IDENTIFIKASI *TWEET CYBERBULLYING* PADA APLIKASI TWITTER MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE* (SVM) DAN *INFORMATION GAIN* (IG) SEBAGAI SELEKSI FITUR

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ni Made Gita Dwi Purnamasari
NIM: 145150201111005

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGESAHAN

IDENTIFIKASI *TWEET CYBERBULLYING* PADA APLIKASI TWITTER MENGGUNAKAN METODE *SUPPORT VECTOR MACHINE* (SVM) DAN *INFORMATION GAIN* (IG) SEBAGAI SELEKSI FITUR

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh:

Ni Made Gita Dwi Purnamasari

NIM: 145150201111005

Skripsi ini telah diuji dan dinyatakan lulus pada

05 Juli 2018

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
M. Ali Fauzi, S.Kom, M.Kom	Indriati, S.T, M.Kom
NIK: 201502 890101 1 001	NIP: 19831013 201504 2 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 05 Juli 2018

Jade Gita Dwi Purnamasari

NIM: 145150201111005

KATA PENGANTAR

Dalam pelaksanaan dan penulisan laporan skripsi ini penulis mendapatkan banyak bantuan dan dukungan dari banyak pihak baik secara moril maupun material. Pada kesempatan ini, penulis ingin menyampaikan terimakasih yang sebesar-besarnya kepada:

1. Bapak M. Ali Fauzi, S.Kom, M.Kom dan Ibu Indriati, S.T, M.Kom, selaku Dosen Pembimbing yang telah membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini dengan baik.
2. Ibu Liana Shinta selaku Dosen Bahasa Indonesia yang telah membantu memvalidasi data skripsi penulis.
3. Seluruh Dosen FILKOM UB yang telah membagi ilmunya kepada penulis selama masa perkuliahan dan Civitas Akademik FILKOM UB yang telah memberikan dukungan dan bantuan selama menempuh studi dan penyelesaian skripsi ini.
4. Kedua Orang Tua penulis dan seluruh keluarga besar atas segala doa, nasihat, perhatian, dukungan dan kasih sayang yang senantiasa diberikan kepada penulis demi terselesaikannya skripsi ini.
5. Sahabat Reti, Ema, Kholi, Yessi dan semua sahabat Informatika Kelas A angkatan 2014, terima kasih atas segala bantuan dan dukungannya selama menempuh studi di FILKOM UB.

Terlepas dari semua itu, penulis menyadari bahwa masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu penulis menerima segala saran dan kritik membangun dari pembaca untuk menyempurnakan skripsi ini. Semoga laporan skripsi ini memberikan manfaat bagi semua pihak.

Malang, 05 Juli 2018

Penulis

workpurnamasari@gmail.com

ABSTRAK

Ni Made Gita Dwi Purnamasari, Identifikasi *Tweet Cyberbullying* pada Aplikasi Twitter menggunakan Metode *Support Vector Machine* (SVM) dan *Information Gain* (IG) sebagai Seleksi Fitur

Dosen Pembimbing: M. Ali Fauzi, S.Kom, M.Kom dan Indriati, S.T, M.Kom

Cyberbullying merupakan salah satu tindakan yang melanggar UU ITE dimana kejahatan ini dilakukan di media sosial salah satunya aplikasi Twitter. Tindakan ini sulit terdeteksi jika tidak ada yang me-*report tweet* tersebut. Identifikasi *tweet cyberbullying* bertujuan untuk mengklasifikasikan *tweet* yang mengandung konten *bullying* pada akun Bapak Fadli Zon selaku Wakil Ketua DPR RI. Klasifikasi dilakukan dengan menggunakan metode *Support Vector Machine* (SVM) dimana metode ini akan mencari *hyperplane* pemisah antara kelas negatif dan positif. Penelitian ini merupakan klasifikasi teks dimana semakin banyak datanya semakin banyak fitur yang dihasilkan, oleh karena itu penelitian ini juga menggunakan seleksi fitur *Information Gain* (IG) untuk menyeleksi fitur yang tidak relevan terhadap klasifikasi. Dimana nilai *information gain* akan diurutkan secara *ascending* untuk fitur diseleksi berdasarkan nilai *threshold* yang telah ditentukan. Nilai *information gain* akan bernilai tinggi jika term merepresentasikan satu kelas dengan hanya muncul sekali atau lebih pada kelas tersebut. Proses sistem dimulai dari *text preprocessing* dengan tahapan tokenisasi, *filtering*, *stemming* dan pembobotan kata. Kemudian melakukan seleksi fitur *information gain* dengan menghitung nilai *entropy* tiap kata. Setelah itu melakukan proses klasifikasi berdasarkan fitur yang telah diseleksi dan hasil keluaran sistem berupa identifikasi apakah *tweet* termasuk *bully* atau bukan *bully*. Hasil akurasi menggunakan metode SVM dengan *accuracy* 75%, *precision* 70.27%, *recall* 86.66% dan *f-measur*e 77.61% pada percobaan nilai *iterMax* = 20, $\lambda = 0.5$, $\gamma = 0.001$, $\varepsilon = 0.000001$, dan C = 1. Dan nilai *threshold* seleksi fitur *information gain* terbaik didapatkan adalah 90%, dengan nilai *accuracy* 76.66%, *precision* 72.22%, *recall* 86.66% dan *f-measure* 78.78%.

Kata kunci: *Cyberbullying*, Klasifikasi, *Support Vector Machine*, *Information Gain*.

ABSTRACT

Ni Made Gita Dwi Purnamasari, Cyberbullying Tweet Identification on Twitter using Support Vector Machine and Information Gain as Feature Selection

Supervisors: M. Ali Fauzi, S.Kom, M.Kom and Indriati, S.T, M.Kom

Cyberbullying is one of the actions that violate the ITE Law where the crime is committed on social media applications such as Twitter. This action is difficult to detect if no one is reporting the tweet. Cyberbullying tweet identification aims to classify tweets containing bullying on Mr Fadli Zon account who is the Vice Chairmaman of the Indonesian Parliament. Classification is done using Support Vector Machine method where this method aims to find the dividing hyperplane between negative and positive class. This study is a text classification where more data is used, the more features are produced, therefore this research also uses Information Gain as feature selection to select features that are not relevant to the classification. The entropy of feature will be sorted ascending for the selected feature based on the predefined threshold value. The entropy will be of high value if the term represents a class by appearing only ne or more in that class. The process of the system starts from text preprocessing with tokenizing step, filtering, stemming and term weighting. Then perform the information gain feature selection by calculating the entropy value of each term. After that perform the classification process based on the features that have been selected, and the output of the system in the form of identification whether the tweet is bullying or not. The result of accuracy using SVM method with accuracy 75%, precision 70.27%, recall 86.66% and f-measure 77.61% on experiment iterMax value = 20, λ = 0.5, γ = 0.001, ε = 0.000001, and C = 1. And threshold value the best information gain feature selection is 90%, with accuracy value 76.66%, precision 72.22%, recall 86.66% and f-measure 78.78%

Keywords: *Cyberbullying*, *Classification*, *Support Vector Machine*, *Information Gain*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

BAB 1 PENDAHULUAN

1.1 Latar belakang

Di jaman teknologi yang berkembang ini, sosial media bagaikan suatu kebutuhan di masyarakat. Dengan sosial media masyarakat bisa berkomunikasi dengan orang terdekat dan atau orang yang tidak dikenal. Tetapi belakangan ini sosial media telah menjadi tempat untuk mengujar kebencian dan menyebarkan *hoax*/sara. Walaupun tidak semua pengguna menggunakan sosial media untuk melakukan hal tersebut. Twitter adalah salah satu sosial media yang saat ini digandrungi oleh semua kalangan dimulai dari remaja sampai orang tua. Twitter itu sendiri merupakan layanan jejaring sosial yang penggunanya dapat mengirim dan membaca pesan teks atau yang dikenal sebagai kicauan atau *tweet*.

Di Twitter pengguna bisa saling membagi informasi dan memposting kegiatan mereka. Tetapi banyak juga pengguna yang menyebarkan komentar-komentar negatif mereka terhadap suatu postingan. *Cyberbullying* merupakan perlakuan kasar yang dilakukan di jejaring sosial. Biasanya hal tersebut dilakukan karena pelaku dendam, iri, marah, sakit hati atau hanya memang karena untuk menjelekan korbannya tersebut. Walau terkadang pelaku juga tidak mengenal korbannya tersebut. Menurut survey oleh *University of British Colombia* dari 733 peserta terdapat ada 25-30% dari peserta survey yang pernah di-*bully* di media sosial, dan ada 12% dari peserta survey mengaku bahwa pernah menjadi pem-*bully. Cyberbullying* bisa mempengaruhi mental dari korban tersebut, dan *cyberbullying* merupakan hal yang marak terjadi di media sosial Twitter pada saat ini (Noviantho, et al., 2017).

Penyalahgunaan tentang sosial media diatur pada Undang-Undang Informasi dan Transaksi Elektronik (UU ITE) nomor 11 tahun 2008 pasal 27 ayat 3 yang menyatakan tentang penyebaran nama baik atau penghinaan. Dan *cyberbullying* termasuk ke dalam kategori tersebut dimana terdapat penghinaan. Saat ini KOMINFO tengah bekerja sama dengan Google dan Twiiter dalam pemberantasan konten-konten negatif di internet seperti pornografi, *hoax*, *cyberbullying*, dan lain-lain. KOMINFO berharap agar pencegahan penyebaran konten negatif tersebut cepat diselesaikan. Pihak Twitter pun sudah menyediakan fungsi dimana report tentang konten negatif tersebut secara tersendiri agar lebih cepat ditanggapi (Devaga, 2017). Oleh karena itu dengan adanya sistem otomatis untuk identifikasi *tweet cyberbullying* dapat membantu dalam penanganan *tweet* pengujar kebencian lebih cepat diatasi dan lebih efisien.

Penelitian oleh Miftah Andriansyah tentang *cyberbullying* yang berjudul *Cyberbullying Comment Classification on Indonesian Selebgram Using Support Vector Machine (SVM) Method* dimana *cyberbullying* di fokuskan pada *comment* sebuah akun seorang selebgram (selebritis Instagram) di aplikasi Instagram. Data yang digunakan diambil dari *comment section* salah satu foto yang diunggah oleh

akun selebgram Awkarin, dan hasil akurasi yang didapatkan cukup tinggi. Masalah dalam penelitian klasifikasi ini adalah ketika bahasa dari *comment* yang digunakan tidak termasuk dalam *bullying*, tetapi dalam artinya *comment* tersebut termasuk dalam bullying (Adriansyah, et al., 2017).

Penelitian tentang *cyberbullying* (Noviantho, et al., 2017) yang berjudul *Cyberbullying Classification using Text Mining,* peneliti menggunakan metode *Support Vector Machine* (SVM) dan *Naive Bayes* untuk proses klasifikasi. Data yang digunakan didapatkan dari aplikasi Kaggle (www.kaggle.com), dimana aplikasi tersebut menyediakan total percakapan 1600 dari website Formspring.me yang menggunakan Bahasa Indonesia. Peneliti mengolah data dengan teknik *Preprocessing*, *Extraction*, *Classification* dan *Evaluation*. Peneliti juga membandingan hasil dari penelitian sebelumnya oleh Kelly Reynolds tahun 2012 yang menggunakan *Decision Tree* dan *K-Nearest Neighbor* (K-NN). Dari penelitian tersebut didapatkan hasil bahwa metode SVM lebih baik dalam pengklasifikasian *cyberbullying* dibanding metode K-NN dan *Decision Tree* yang digunakan oleh Kelly Reynolds.

Dalam komputasinya penggunaan metode SVM memakan waktu yang cukup lama tergantung pada banyak fitur yang digunakan. Dalam pengklasifikasian teks, setiap kata dari hasil pemrosesan teks akan dijadikan fitur, dengan begitu jika menggunakan dokumen yang banyak akan menghasilkan fitur yang banyak pula. Oleh karena itu seleksi fitur digunakan dalam penelitian ini untuk menyeleksi fitur-fitur yang relevan saja agar dapat mengurangi waktu dari komputasi metode SVM.

Pada penelitian tentang Komparasi Algoritma Klasifikasi *Machine Learning* dan *Feature Selection* pada Analisis Sentimen Review Film (Chandani, et al., 2015) menggunakan metode *Artificial Neural Network* (ANN), *Support Vector Machine* (SVM) dan *Naive Bayes* (NB), serta menggunakan seleksi fitur *Information Gain* (IG), *Chi Square, Forward Selection* dan *Backward Selection.* Data yang digunakan diambil dari situs *Internet Movie Database* (IMDb) berupa *review* film dari *user*. Dari penelitian tersebut didapatkan hasil bahwa metode SVM memiliki akurasi paling tinggi dibanding metode klasifikasi yang lainnya. Serta fitur seleksi IG menghasilkan akurasi yang lebih tinggi dibanding fitur seleksi yang lain yang digunakan pada metode SVM.

Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon dan Vapnik pada tahun 1992 di *Annual Workshop on Computational Learning*. Konsep dasar SVM adalah gabungan dari teori-teori komputasi yang sudah ada. SVM merupakan salah satu teknik yang digunakan untuk melakukan prediksi dan klasifikasi. Ada beberapa fungsi *kernel* untuk pengklasifikasian SVM, salah satunya yaitu SVM *Polynomial*. SVM *Polynomial* diketahui memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan metode lainnya dalam hal pengklasifikasian, ini dibuktikan dari penelitian (Noviantho, et al., 2017) dimana metode SVM *Polynomial* memiliki nilai akurasi tertinggi yaitu 99.41%.

Pada penelitian ini metode yang digunakan adalah *Support Vector Machine* (SVM), dimana metode ini sudah banyak terbukti pada pengujian penelitian

sebelumnya dapat menghasilkan nilai akurasi yang tinggi. Pemilihan metode SVM karena memiliki tingkat akurasi yang lebih tinggi dibanding dengan metode klasifikasi yang lainnya. Tetapi dalam komputasi metode SVM tergolong memakan waktu yang cukup lama, maka dari itu peneliti menggunakan seleksi fitur untuk mengurangi fitur-fitur agar memperingan kerja komputasi dari metode SVM. Fitur seleksi yang digunakan adalah *Information Gain* (IG), dimana IG digunakan untuk menentukan batas dari sebuah fitur. Banyaknya fitur akan mempengaruhi proses komputasi dan jika banyak fitur yang tidak relevan digunakan dalam klasifikasi maka dapat mempengaruhi hasil akurasi.

1.2 Rumusan masalah

Berdasarkan uraian pada latar belakang, dapat dirumuskan suatu permasalahan :

1. Bagaimana pengaruh parameter pada metode *Support Vector Machine* (SVM) terhadap hasil identifikasi *tweet cyberbullying*?
2. Bagaimana pengaruh penggunaan fitur seleksi *Information Gain* (IG) terhadap hasil identifikasi *tweet cyberbullying* dengan metode *Support Vector Machine* (SVM)?

1.3 Tujuan

Tujuan dari penelitian ini adalah membangun sebuah sistem yang dapat mengidentifikasi *tweet* (kicauan) pada aplikasi Twitter yang mengandung pesan *cyberbullying*. Serta mengetahui pengaruh parameter metode *Support Vector Machine* dan penggunaan seleksi fitur *Information Gain* terhadap hasil identifikasi *tweet cyberbullying.*

1.4 Manfaat

Manfaat yang diharapkan dari hasil penelitian ini adalah sistem ini dapat membantu pengguna untuk memilah-milah *tweet* (kicauan) yang berbau negatif atau *cyberbullying* di aplikasi Twitter untuk di-*report* dan dihapus.

1.5 Batasan masalah

Dari permasalahan yang telah dijelaskan diatas, maka berikut ini diberikan batasan permasalahn tersebut, diantaranya :

1. Data *Tweet* yang digunakan mengenai wakil ketua DPR Bapak Fadli Zon
2. Data *Tweet* diperoleh dari *Application Programming Interface* (API) Twitter.
3. Jumlah data yang digunakan sebanyak 300 *tweet*.
4. Sistem ini dibangun menggunakan bahasa pemrograman Java.
5. Keluaran sistem ini berupa hasil identifikasi apakah *tweet* tersebut mengandung konten *cyberbullying* atau tidak.

1.6 Sistematika Pembahasan

Penelitian ini disusun berdasarkan sistematika penulisan :

BAB I PENDAHULUAN

Bab ini berisi Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan Penelitian, Manfaat Penelitian, dan Sistematika Pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi teori-teori yang berhubungan dengan twitter dan *cyberbullying* serta metode *Support Vector Machine* (SVM), TF-IDF, dan seleksi fitur *Information Gain* (IG). Serta penelitian-penelitian yang berhubungan dengan penelitian ini yang pernah dilakukan sebelumnya.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan metode yang digunakan dan langkah kerja yang dilakukan dalam penulisan yang terdiri dari Tipe penelitian, strategi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumpulan data, implementasi algoritma, dan jadwal penelitian.

BAB IV PERANCANGAN

Bab ini berisi perancangan algoritme dan manualisasi perhitungan data menggunakan metode *Support Vector Machine* (SVM) dan seleksi fitur *Information Gain* (IG) beserta perancangan pengujian yang akan dilakukan pada bab berikutnya.

BAB V HASIL

Pada bab ini berisi hasil dari implementasi metode *Support Vector Machine* (SVM) dan seleksi *Information Gain* beserta penjelasan dari kode program implementasi sistem.

BAB VI PEMBAHASAN

Pada Bab ini melakukan pengujian sistem dan makna dari hasil yang telah diperoleh untuk menjawab pertanyaan dan masalah penelitian, serta menjelaskan pemahaman baru yang didapatkan dari hasil penelitian, yang nantinya diharapkan berguna dalam pengembangan keilmuan.

BAB VII PENUTUP

Bab ini berisi kesimpulan yang diambil dari hasil penelitian ini dan saran yang ditujukan kepada pembaca dan peneliti selanjutnya tentang kekurangan dari penelitian ini agar bisa dikembangkan untuk selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian ini didasarkan pada studi literatur buku dan beberapa penelitian sebelumnya yang pernah dilakukan yang berkaitan dengan twitter dan klasifikasi penggunaan metode *Improved* K-NN. Publikasi penelitian terdahulu dalam bentuk paper, jurnal, kripsi, tesis, maupun seminar international digunakan sebagai sumber kajian pustakan dalam penelitian ini. Studi literatur dilakukan dengan tujuan untuk memperkuat pemahaman domain permasalahan dan juga mencari penyelesaian masalah terbaik.

Pada penelitian Penerapan Sentimen Analisis Acara Televisi pada Twitter Menggunakan *Support Vector Machine* dan Algoritma Genetika sebagai Metode Seleksi Fitur (Darma, 2018) dimana sistem melakukan perhitungan *rating* berdasarkan *tweet* positif dan negatif pada suatu acara televisi. Data tweet untuk sentiment analisis didapatkan dari API Twitter. Dari hasil pengujian didapatkan rata-rata nilai *error* sebesar 0.562 untuk rentang nilai 0-5. Dengan diterapkannya algoritma genetika sebagai seleksi fitur didapatkan nilai *error* 0.62 dimana perbedaan akurasi yang didapat sebelum dan sesudah diterapkannya seleksi fitur.

Klasifikasi dengan metode *Support Vector Machine* (SVM) yang dilakukan (Putranti & Winarko, 2014) yang berjudul Analisis Sentimen Twitter untuk Teks Berbahasa Indonesia dengan *Maximum Entropy* dan *Support Vector Machine* merupakan penelitian untuk mengklasifikasian sentimen menjadi kelas sentimen positif dan negatif. Data yang digunakan diambil berdasarkan *query* dan *term* objek pada aplikasi yang terhubung dengan Twitter API. Kemudian dilakukan proses TF-IDF untuk pemrosesan teks dan pembobotan entropi. SVM dipilih karena kemampuannya dalam generalisasi pengklasifikasian suatu *pattern*. Hasil akurasi yang didapatkan dengan menggunakan SVM adalah sebesar 86.81% dengan waktu proses 1688 detik pada tipe *kernel* sigmoid.

Penelitian tentang *cyberbullying* oleh (Adriansyah, et al., 2017) yang berjudul *Cyberbullying Comment Classification on Indonesian Selebgram Using Support Vetctor Machine Method* dimana cyberbullying di fokuskan pada *comment* sebuah akun seorang selebgram (selebritis Instagram) di aplikasi Instagram. Ada 1053 data *comment* yang digunakan untuk pengklasifikasian dengan metode SVM, dan hasil akurasi yang didapatkan adalah sebesar 79.412%. Akurasi bisa ditingkatkan dengan menaikan jumlah data *training* dan mengembangkan *comment* semantik. Bahasa dari *comment* yang digunakan tidak termasuk dalam *bullying*, tetapi dalam artinya *comment* tersebut termasuk dalam bullying.

Pada penelitian klasifikasi *cyberbullying* (Noviantho, et al., 2017), peneliti membandingkan hasil klasifikasi cyberbullying dari penelitian sebelumnya oleh Kelly Reynolds (2012). Dimana Reynolds menggunakan metode *K-Nearest Neighbor* (K-NN) dan *Decision Tree* untuk pengklasifikasian. Data yang digunakan

merupakan data percakapan berbentuk tanya jawab sesuai dengan *fields* yang ada di website Formspring.com. Hasil pengujian yang didapatkan menunjukan bahwa pengklasifikasian *cyberbullying* menggunakan metode SVM memiliki nilai akurasi yang tinggi dengan rata-rata SVM-Linear 99.04%, SVM-Poly 99.41%, SVM-rbf 63.77%, SVM-sigmoid 99.04%, dan *Naive Bayes* 96.98%. sedangakan rata-rata dari nilai akurasi oleh penelitian Reynolds yaitu dengan *decision tree* 78.28%, K-NN (k=1) 89.01%, dan K-NN (k-3) 74.53%.

Komparasi Algoritma Klasifikasi *Machine Learning* (Chandani, et al., 2015) tentang analisis sentimen untuk review film menggunakan beberapa algoritma klasifikasi yaitu *Naive Bayes* (NB), *Support Vector Machine* (SVM), dan *Artificial Neural Network* (ANN), serta menggunakan beberapa *feature selection* seperti *Information Gain, Chi Square, Forward Selection, dan Backward Elimination*. Data yang digunakan diambil dari situs *Internet Movie Database* (IMDb). Dimana hasil yang didapatkan terlampir pada tabel 2.1 dan tabel 2.2.

Tabel 2.1 Komparasi Akurasi dan AUC Algoritma Klasifikasi

Algoritma	Akurasi	AUC
ANN	51.80%	0.500
SVM	81.10%	0.904
NB	74.00%	0.734

Tabel 2.2 Komparasi Akurasi dan AUC Algoritma *Feature Selection*

	Information Gain		***Chi Square***		***Forward Selection***		***Backward Elimination***	
	Top K (K = 200)		Top K (K = 100)					
	Akurasi	AUC	Akurasi	AUC	Akurasi	AUC	Akurasi	AUC
ANN	91.40%	0.914	79.60%	0.900	75.50%	0.781	70.20%	0.724
SVM	81.50%	0.929	80.80%	0.853	67.67%	0.698	79.25%	0.844
NB	80.80%	0.853	80.30%	0.867	79.00%	0.807	71.25%	0.689
Average	84.57%	0.899	80.23%	0.873	74.06%	0.762	73.57%	0.752

Berdasarkan dari kesimpulan dalam penelitian sebelumnya, maka penelitian ini akan melakukan identifikasi *tweet cyberbullying* degan menggunakan metode *Support Vector Machine* (SVM) dan seleksi fitur *Information Gain* (IG) agar mendapatkan akurasi yang maksimal dan nilai *Area Under Cover* (AUC) yang didapatkan lebih besar dari metode yang lainnya. Dimana fungsi *kernel* pada metode SVM yang digunakan adalah SVM *Polynomial*. Dengan begitu diharapkan agar mendapatkan hasil akurasi yang lebih baik.

2.2 Twitter

Twitter merupakan salah satu layanan jejaring sosial yang masuk kedalam *Microblogging* atau ngeblog secara singkat dalam satu paragraf dengan maksimal 140 huruf, namun sekarang panjang *tweets* di twitter sudah ditambahkan menjadi 280 huruf. Twitter saat ini merupakan salah satu aplikasi yang digemari oleh semua kalangan dimulai dari orang tua sampai remaja dan anak-anak. Twitter menjadi salah satu alat interaksi bagi penggunanya untuk membagi informasi, atau hanya memposting apa yang sedang kita lakukan saat itu. Sesama pengguna bisa saling me-retweet dan mengomentari postingan dari pengguna lain. Sesama pengguna pun bisa saling mengikuti pengguna lainnya, jadi kita bisa mendapatkan update dari pengguna yang telah diikuti atau biasa disebut dengan *follow* (Motivasee, 2017).

2.3 Cyberbullying

Cyberbullying merupakan salah satu tindak kekerasan yang dilakukan oleh seseorang terhadap korbannya di internet, dimana korban dihina, diejek, dipermalukan dan diintimidasi oleh pelaku. *Cyberbullying* bisa berdampak pada mental korban, bahkan banyak dari korban *bullying* berakhir dengan bunuh diri karena tidak tahan dengan banyak tekanan (Oktaviani, 2013). *Cyberbullying* juga berdampak kepada pelaku karena cyberbullying merupakan salah satu tindakakn yang melawan hukum dan ada ancaman hukuman penjaranya. Diatur pada UU ITE pasal 28 yang menyatakan bahwa segala bentuk penghinaan dan penyebaran nama baik melalu media elektronik akan dikenakan hukuman penjara dan denda sesuai dengan aturan yang berlaku.

Tindakan *cyberbullying* di sosial media tidak mengenal jenis kelamin, beberapa penelitian menunjukan bahwa siapa saja bisa menjadi koban dari *cyberbullying* baik itu laki-laki maupun perempuan. Seseorang yang menjadi korban *cyberbullying* biasanya juga adalah korban *bully* di sekolahnya (Patchin & Hinduja, 2012)Para pelaku *bully* merupakan orang-orang yang cenderung agresif dan sering melanggar aturan. Pada penelitian cyberbullying (Patchin & Hinduja, 2012) mengatakan bentuk-bentuk cyberbullying dilakukan pada media sosial berupa : pemberian nama negatif (nama hewan dan panggilan fisik), penyebaran foto, mengancam, dan pendapat yang merendahkan.

2.4 *Text Preproccesing*

Text preproccesing merupakan salah satu proses pengolahan teks dimana terdiri dari tokenisasi, *filtering*, stemming dan *weighting*. Pengertian *text preprocessing* sendiri adalah suatu proses pengolahan atau pengubahan data yang belum terstruktur menjadi terstruktur sesuai dengan kebutuhan atau proses *mining* lebih lanjut. Pemrosesan ini mengubah data tekstual asli dalam struktur data *mining*, dimana fitur teks yang paling signifikan yang berfungsi untuk membedakan antara kategori teks yang diidentifikasi. Tujuan utama *text preprocessing* adalah untuk mendapatkan fitur utama atau istilah kunci dari

dokumen teks dan untuk meningkatkan relevansi antara kata dan dokumen serta relevansi antara kata dan kategori (Gaigole, et al., 2013)

2.4.1 Tokenisasi

Langkah pertama dalam *text preprocessing* adalah tokenisasi. Tokenisasi merupakan suatu proses pemotongan *string*/kata pada suatu kalimat dan semua tanda baca dan tanda hubung akan dihilangkan. Proses ini bertujuan untuk memisahkan tiap kata agar dapat membedakan karakter-karakter tertentu yang diperlakukan sebagai pemisah kata atau bukan. Proses tokenisasi mengandalkan karakter spasi pada dokumen sebagai pemisah kata (Garcia, 2005).

2.4.2 Filtering

Tahap *filtering* merupakan tahap mengambil kata-kata penting dari sebuah hasil token. Banyak kata yang paling sering digunakan dalam Bahasa Indonesia tidak berguna dalam *Information Retrieval* (IR) dan *text mining,* kata-kata ini disebut *Stopwords* (Gaigole, et al., 2013). Langkah selanjutnya yang dilakukan untuk teks *preprocessing* adalah proses pembuangan kata tidak penting sesuai dengan daftar kata pada *stopwords.* Kamus Stopword yang digunakan didapatkan dari Tala. Misalnya adalah kata-kata “yang”, “di”, dan yang lainnya.

2.4.3 Stemming

Stemming adalah tahap mencari kata dasar dari hasil *filtering*. Teknik *stemming* digunakan untuk mengetahui akar sebuah kata. *Stemming* dilakukan selain untuk memperkecil jumlah indeks berbeda dari suatu dokumen, juga dilakukan untuk mengempokkan kata-kata yang memiliki kata dasar dan arti yang serupa tetapi memiliki bentuk yang berbeda karena terdapat imbuhan yang berbeda (Gaigole, et al., 2013).

Teknik *stemming* yang dilakukan menggunakan *stemming* dari Nazief & Adriani. Teknik *stemming* Nazief & Adriani merupakan algoritma *stemming* untuk teks berbahasa Indonesia, dimana *stemming* Nazief & Adriani memiliki akurasi yang lebih tinggi dibanding algoritme Porter walaupun memiliki waktu komputasi yang lebih lama (Agusta, 2009). Tahapan proses *stemming* Nazief & Adriani sebagai berikut:

1. Mencari kata yang akan di *stem* dalam kamus.
2. Membuang *Inflection Suffixes*, seperti “-lah”, “-kah”, “-ku”, atau “-nya”.
3. Menghapus *Derivation Suffixes*, seperti “-i”, “-an”, atau “-kan”.
    a. Jika akhiran ”-an”dihapus dan huruf terakhir dari kata adalah “-k” maka huruf “-k” juga ikut dihapus.
4. Menghapus Derivation Prefix.
    a. Memeriksa table kombinasi awalan-akhiran yang tidak diijinkan yang ditunjukan pada Tabel 2.3. Jika ditemukan maka berhenti.

Tabel 2.3 Kombinasi Awalan-Akhiran yang Tidak Diijinkan

Awalan	Akhiran yang tidak diijinkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i. -kan

5. Melakukan Recoding, kemudian selesai.

2.4.4 Term Weighting

Setelah melakukan proses tokenisasi, *filtering* dan *stemming*, langkah yang dilakukan selanjutnya adalah proses penghitungan TF-IDF (*Term frequency-Inverse Document Frequency*) dimana disini dilakukan proses penghitungan jumlah bobot tiap token hasil stemming yang nantinya akan dibandingkan dengan dokumen lain untuk membandingkan frequency dari jumlah kemunculan token tersebut. Rumus TF-IDF dapat dilihat dalam Persamaan (2.1) dan Persamaan (2.2).

$$w_{t,d} = \begin{cases} 1 + log_{10} tf_{t,d}, & if\ tf_{t,d} > 0 \\ 0, & if\ tf_{t,d} = 0 \end{cases} \quad (2.1)$$

Dimana *tf* adalah *term frequency* yang menyatakan berapa banyak jumlah suatu term dalam sebuah dokumen.

$$idf_t = log_{10}(\frac{N}{df_t}) \quad (2.2)$$

Dimana N merupakan jumlah banyak dokumen, karena terkadang suatu term muncul di beberapa dokumen sehingga proses pencarian *term* unik akan terganggu. IDF befungsi untuk mengurangi bobot suatu term jika kemunculannya dari term tersebut banyak dan tersebar diseluruh dokumen (Gaigole, et al., 2013).

2.5 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) dikenal sebagai teknik pembelajaran (*machine learning*) yang baik. Pembelajaran dilakukan dengan menggunakan data *input* dan data *output*. Pembelajaran dengan cara ini disebut dengan *supervised learning*. Dengan pembelajaran terarah ini dapat diperoleh fungsi yang menggambarkan bentuk ketergantungan antara *input* dan *output*. Metode SVM itu sendiri dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali dipresentasikan tahun 1992 pada Annual Workshop on Computational Learning Theory (Nugroho, et al., 2003). Konsep klasifikasi dengan SVM adalah mencari *hyperplane* (garis) terbaik yang berfungsi sebagai pemisah dua kelas data. SVM memaksimalkan *margin*, yang merupakan jarak pemisah antara kelas data. SVM juga mampu bekerja pada dataset yang berdimensi tinggi dengan menggunakan

kernel trick. Ada beberapa macam fungsi *kernel* SVM, yaitu : Linear, *Polynomial*, Gaussian RBF, Sigmoid, Invers Multi Kuadratik, dan *Additive*.

Pada penelitian ini fungsi kernel yang digunakan adalah SVM *Polynomial*. SVM linear digunakan ketika data yang akan diklasifikasi dapat terpisah dengan sebuah *hyperplane*, sedangkan SVM non-linear digunakan ketika data hanya dapat dipisahkan dengan garis lengkung. SVM *Polynomial* memiliki definisi fungsi dengan Persamaan (2.3).

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d \quad (2.3)$$

Dimana $K(\overrightarrow{x_i}, \overrightarrow{x_j})$ merupakan fungsi *kernel*, x merupakan fitur dan d merupakan ordo.

2.5.1 *Sequential Training* pada SVM

Hyperplane dalam SVM yang optimal didapatkan dengan merumuskannya ke dalam QP problem dan diselesaikan menggunakan *library* yang banyak tersedia dalam analisis *numeric*. Namun ada alternatif lain yang cukup sederhana yaitu metode sequential. Metode ini dikembangkan oleh Vijayakumar untuk mencari nilai α, yang diuraikan dalam tahapan:

1. Inisialisasi $\alpha_i = 0$

   Menghitung nilai matriks *Hessian* dengan menggunakan persamaan (2.4).

   $$D_{ij} = y_i y_j (K(x_i . x_j) + \lambda^2) \quad (2.4)$$

   Dimana y merupakan kelas dari data ke-i dan ke-j, $K(x_i . x_j)$ merupakan fungsi kernel polynomial yang digunakan.

2. Menghitung setiap level dengan tahapan menggunakan persamaan (2.5) sampai (2.7).

   a) $E_i = \sum_{j=1}^{n} \alpha_i D_{ij}$ (2.5)

   b) $\delta\alpha_i = \min\{\max[\gamma(1 - E_i), -\alpha_i], C - \alpha_i\}$ (2.6)

   c) $\alpha_i = \alpha_i + \delta\alpha_i$ (2.7)

3. Melakukan perulangan ke tahap 2 sampai nilai α mencapai konvergen.

Dimana ϒ merupakan parameter untuk mengontrol kecepatan proses learning. Konvergensi dapat didefinisikan dari perubahan nilai α (Nugroho, et al., 2003).

2.6 *Feature Selection*

Feature Selection atau dalam Bahasa Indonesia seleksi fitur merupakan salah satu proses pemilihan fitur yang relevan dalam hal masalah pembelajaran target. Tujuan dari seleksi fitur adalah untuk menghapus fitur yang berlebihan dan yang tidak relevan (Bangsheng, 2013). Fitur yang tidak relevan adalah yang tidak memberikan informasi bermanfaat tentang data, dan fitur yang berlebihan

merupakan fitur yang memberikan informasi lebih banyak daripada fitur yang saat ini dipilih. Dengan kata lain, fitur berlebihan memberikan informasi yang berguna tentang kumpulan data, namun informasinya telah disediakan oleh fitur yang sudah dipilih sebelumnya.

2.7 *Information Gain* (IG)

Information Gain (IG) merupakan salah satu metode untuk seleksi fitur yang banyak digunakan oleh peneliti untuk menentukan batas dari kepentingan sebuah atribut (Deng & Runger, 2012). IG banyak digunakan karena memiliki nilai akurasi yang lebih tinggi dibandingkan dengan seleksi fitur yang lainnya. Proses seleksi fitur untuk membuang atribut yang tidak relevan, hal ini dilakukan untuk memungkinkan algoritma klasifikasi bekerja lebih cepat dan efektif dan dapat meningkatkan akurasi suatu algoritma klasifikasi. Nilai IG diperoleh dari nilai *entropy* sebelum pemisahan dikurang dengan nilai *entropy* setelah pemisahan. Nilai ini digunakan untuk penentuan atribut mana yang akan dibuang atau digunakan. Atribut yang memenuhi kriteria pembobotan nantinya akan digunakan untuk proses klasifikasi.

Dalam pemilihan fitur dengan IG dilakukan dengan 3 tahapan, yaitu :

1. Menghitung nilai IG untuk setiap atribut.
2. Menentukan *threshold* (batas). Hal ini digunakan untuk menentukan atribut yang bobotnya lebih kecil dari *treshold* akan dibuang.
3. Memperbaiki dataset dengan pengurangan atribut.

Seleksi fitur *Information Gain* $IG(t)$ dirumuskan pada persamaan (2.8).

$$IG(t) = -\sum_{i=1}^{|C|} P(C_i) \log P(C_i) + P(t) \sum_{i=1}^{|C|} P(C_i|t) \log P(C_i|t) + P(\bar{t}) \sum_{i=1}^{|C|} P(C_i|\bar{t}) \log P(C_i|\bar{t}) \quad (2.8)$$

Dimana C_i merupakan kelas data, $P(C_i)$ merupakan peluang dari kelas data, $P(t)$ dan $P(\bar{t})$ merupakan peluang *term* t yang muncul atau tidak muncul dalam dokumen. Dalam machine *learning*, perolehan informasi dapat digunakan untuk membantu menentukan peringkat fitur (Bangsheng, 2013).

2.8 Evaluasi

Hasil identifikasi *tweet cyberbullying* dapat diuji dengan menggunakan beberapa teknik pengujian yaitu *accuracy classification*, *precision*, *recall* dan *f-measure*. *Accuracy classification* merupakan suatu nilai tingkat kedekatan antara nilai yang diprediksi dengan nilai aktual. *Precision* merupakan tingkat ketepatan antara informasi yang diminta *user* dengan jawaban yang diberikan oleh sistem, sedangkan *recall* merupakan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Dan *f-measure* merupakan gabungan dari *precision* dan *recall*. Nilai *precision* dan *recall* terhadap suatu keadaan dapat memiliki bobot yang berbeda. Ukuran yang menampilkan timbal balik antara *precision* dan *recall* adalah *f-measure*, yang merupakan nilai *harmonic mean* dari *precision* dan *recall*

(putubuku, 2008). Adapun persamaan dari pengujian *accuracy classification, precision, recall* dan *f-measure* terdapat pada persamaan (2.9) sampai (2.12).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.9)$$

$$Precision = \frac{TP}{TP+FP} \quad (2.10)$$

$$Recall = \frac{TP}{TP+FN} \quad (2.11)$$

$$F - measure = 2 \times \frac{precision \times recall}{precision+recall} \quad (2.12)$$

Persamaan (2.5), (2.6), (2.7) dan (2.8) dapat ditentukan menggunakan nilai yang sudah ditentukan pada *confusion matriks*. *Confusion matriks* merupakan klasifikasi aktual dan prediksi pada sistem klasifikasi. Pada penelitian ini terdapat dua kelas klasifikasi, maka evaluasi menggunakan *confusion matriks* 2x2 yang ditunjukan pada Tabel 2.4 berikut:

Tabel 2.4 *Confusion matriks 2x2*

***Confusion Matriks* 2x2**		**Prediksi**	
		Positive	*Negative*
Nilai Sebenarnya	*Positive*	*True Positive* (TP)	*False Negative* (FN)
	Negative	*False Positive* (FP)	*True Negative* (TN)

Dimana pengertian dari istilah-istilah pada Tabel 2.4 untuk evaluasi menggunakan *confusion matriks* 2x2, yaitu:

a. TP merupakan nilai *positive* yang diklasifikasikan *positive.*
b. FN merupakan nilai *positive* yang diklasifikasikan *negative.*
c. FP merupakan nilai *negative* yang diklasifikasikan *positive*.
d. TN merupakan nilai *negative* yang diklasifikasikan *negative*.

BAB 3 METODOLOGI

Metode penelitian yang akan dilakukan dalam penelitian ini dijelaskan dalam beberapa subbab yaitu tipe dari penelitian, strategi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumpulan data, implementasi algoritme, dan jadwal penelitian.

3.1 Tipe Penelitian

Tipe dari penelitian ini adalah nonimplementatif-analitik. Dimana penelitian ini memfokuskan pada analisis terhadap pengaruh dari parameter *Support Vector Machine* (SVM) dan parameter seleksi fitur *Information Gain*. Pendekatan analitik yang dilakukan guna menjelaskan hubungan antar elemen dalam objek penelitian dengan parameter atau situasi yang sedang diteliti.

3.2 Strategi Penelitian

Strategi dalam penelitian ini menggunakan metode eksperimen, yang merupakan suatu penelitian yang berusaha mencari pengaruh parameter tertentu terhadap hasil penelitian yang terkontrol secara ketat dan umumnya dilakukan di laboratorium. Metode eksperimen yang digunakan adalah *machine learning Support Vector Machine* (SVM) dan seleksi fitur *Information Gain*.

3.3 Partisipan Penelitian

Partisipan yang terlibat dalam penelitian ini adalah pengguna media sosial Twitter yang meng-*tweet* tentang wakil ketua DPR (Dewan Perwakilan Rakyat) bapak Fadli Zon yang kontroversial karena *tweet*-nya tentang politik dan pemerintahan. Hasil *tweet* dikategorikan oleh peneliti dan divalidasi oleh Ibu Liana Shinta Dewi, M.A Dosen Bahasa Indonesia.

3.4 Lokasi Penelitian

Penelitian ini dilakukan di ruang Laboratorium Komputasi Cerdas Fakultas Ilmu Komputer Universitas Brawijaya. Penempatan penelitian sesuai dengan studi yang dilakukan oleh peneliti. Alasan pemilihan lokasi penelitian dilakukan untuk mengukur pengaruh penggunaan *Support Vector Machine* (SVM) dan seleksi fitur *Information Gain*.

3.5 Teknik Pengumpulan Data

Pengumpulan data didapatkan dari *Application Programming Interface* (API) Twitter. API sendiri merupakan suatu aplikasi yang disediakan oleh pihak *developer* tertentu agar pihak pengembang lainnya dapat dengan mudah mengakses aplikasi tersebut. Dengan API twitter peneliti dapat mengambil data sesuai dengan kebutuhan penelitian. Pengambilan data menggunakan Rstudio dengan menggunakan bahasa R. Rstudio merupakan salah satu aplikasi *open source* yang ditujukan untuk pengolahan analisis data dan statistik (Nurmansyah,

2015). Penggunaan Rstudio yang mudah dalam mengambil data dari API Twitter dibanding menggunakan bahasa pemrograman yang lain, sehingga mempermudah peneliti dalam mengumpulkan data.

3.6 Implementasi Algoritme

Implementasi algoritme dilakukan dengan mengolah data dokumen kemudian melakukan manualisasi perhitungan dengan metode *Support Vector Machine* (SVM) dan seleksi fitur *Information Gain* (IG) yang digunakan. Manualisasi dimulai dengan melakukan pemrosesan teks dimulai dari tokenisasi, *filtering*, *stemming* serta penghitungan *Term Frequency – Inverse Document Frequency* (TF-IDF), kemudian penghitungan seleksi fitur IG, lalu klasifikasi menggunakan metode SVM. Implmentasi algoritme juga diikuti dengan *flowchart* penggunaan metode SVM dan seleksi fitur IG, serta perancangan pengujian yang akan dilakukan. Setelah itu melakukan koding untuk mengimplementasikan algoritme ke dalam bahasa pemrograman java.

3.7 Analisis Data

Pada penelitian ini dilakukan analisis pada pengujian untuk mengetahui hasil implementasi yang telah dilakukan. Terdapat beberapa pengujian sistem pada penelitian ini yaitu menguji parameter algoritma SVM dan pengujian penggunaan parameter seleksi fitur IG serta akurasi dari sistem. Skenario pengujian yang akan dilakukan antara lain:

1. Pengujan parameter *Sequential Training* SVM.
    a. Pengujian λ (*Lambda*)
    b. Pengujian ϒ (*Gamma*)
    c. Pengujian ε (*Epsilon*)
    d. Pengujian C (*Complexity*)
    e. Pengujian Iterasi Maksimum
2. Pengujian parameter *Threshold Information Gain*

3.8 Jadwal Penelitian

Jadwal penelitian ditetapkan agar penelitian ini tersusun sesuai dengan tahapan penelitian dan selesai pada waktu yang telah ditentukan. Penelitian ini dilaksanakan mulai dari bulan Februari 2018 sampai Mei 2018, dengan jadwal yang terlampir pada tabel 3.1.

Tabel 3.1 Jadwal Penelitian

No	Uraian	Februari				Maret				April				Mei			
		Minggu ke-															
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Kepustakaan																
2	Pengumpulan dan Validasi Data																
3	Implementasi Algoritme																
4	Pengujian dan Analisis																
5	Kesimpulan dan Saran																

BAB 4 PERANCANGAN

Pada bab ini membahas tentang permasalahan dalam penelitian dan perancangan untuk proses penyelesaian serta perancangan pengujian yang akan dilakukan. Di bab ini juga terdapat implementasi dari hasil perancangan yang telah dilakukan.

4.1 Deskripsi Permasalahan

Bullying sekarang banyak terjadi di lingkungan sekitar maupun di sosial media, salah satunya di aplikasi Twitter. *Cyberbullying* termasuk dalam pelanggaran UU ITE 2008 dimana pelaku *bullying* dapat dikenakan hukuman jika terbukti melakukan perbuatan *bullying* dengan sengaja. Di aplikasi Twitter begitu banyak pengguna yang melakukan aktivitas seperti mengunggah status dan mengomentari status atau hanya membaca saja. Adapun pengguna yang mengomentari status pengguna lain dengan mem-*bully* korban tersebut dengan sengaja. Hal ini pun tidak dapat ditindaki dengan cepat jika korban tidak melakukan *report* agar *tweet* tersebut diatasi. Pihak twitter telah menyediakan server tersendiri untuk *report* tweet yang mengandung konten negatif seperti pronografi, terorisme, dan *bullying*. Namun hal itu kurang efisien karena twitter baru dapat menghapus tweet tersebut jika ada yang me-*report*. Dengan adanya sistem otomatis identifikasi *tweet cyberbullying*, maka *tweet-tweet* yang mengandung konten *cyberbullying* dapat ditindaki lebih cepat dan efisien.

4.2 Deskripsi Umum Sistem

Sistem ini merupakan sistem yang dapat menyelesaikan permasalahan proses identifikasi *tweet* yang mengandung konten *bullying*. *Input* dari sistem ini merupakan *tweet* mengenai bapak wakil ketua DPR Fadli Zon. Keluaran dari sistem ini adalah identifikasi apakah *tweet* yang diproses merupakan *tweet cyberbullying*. Sistem menggunakan bahasa pemrograman Java, dan data disimpan dalam file *Microsoft Excel*. Kerja sistem nantinya akan terbagi menjadi dua yaitu, identifikasi menggunakan SVM saja, dan identifikasi menggunakan gabungan dari seleksi fitur *Information Gain* dan SVM. Hal tersebut bertujuan untuk membandingkan pengaruh ketika sistem menggunakan seleksi fitur *Information Gain* dan ketika tidak menggunakan seleksi fitur.

4.3 Manualisasi Perhitungan Data

Manualisasi perhitungan data dimulai dari *Text Preprocessing* kemudian dilanjutkan dengan seleksi fitur *Information Gain* lalu identifikasi menggunakan metode *Support Vector Machine* (SVM). Data *training* yang digunakan ditunjukan pada Tabel 4.1.

Tabel 4.1 Data *Training*

No	Dokumen	Kelas
1	karena @fadlizon kebanyakan nyinyir jadi otaknya ga dipake mikir	Bullying
2	@fadlizon komentar terus aja lu zon, nyinyiranmu bikin eneg	Bullying
3	@fadlizon Goblok, gak mikir lo ye, makin lo nyinyir gak pake otak kaya gini, makin eneg orang liat muke lo! Dasar tukang nyinyir	Bullying
4	pak @fadlizon juga punya prestasi yang membanggakan kok makanya bisa duduk di DPR	No
5	Bangga sama bang @fadlizon mantap, tidak takut	No
6	@fadlizon walaupun anda dihina terus sy tetap bangga sama anda pak dengan prestasi bapak, mantap	No

Kemudian data *testing* yang digunakan untuk pemrosesan teks ditunjukan pada Tabel 4.2.

Tabel 4.2 Data *Testing*

No	Dokumen	Kelas
1	yaelah.. Mulai lagi si @fadlizon nyinyir di publik, malu-maluin goblok	?
2	@fadlizon sabar ya pak.. Prestasi bapak akan mengalahkan omongan mereka, tidak usah khawatir, kami bangga	?

4.3.1 *Text Preprocessing*.

Text Preprocessing dimulai dari tokenisasi yaitu memisahkan kata berdasarakan spasi, kemudian dilanjutkan dengan *filtering* dimana melakukan *stopword* untu membuang kata yang tidak penting. Selanjutnya dilakukan *stemming* yaitu mencari kata dasar dari hasil filtering, dan yang terakhir adalah term weighting yaitu pembobotan tiap kata.

4.3.1.1 Tokenisasi

Perhitungan manualisasi tokenisasi ditunjukan pada Tabel 4.3 untuk data *training*. Tiap token dipisahkan oleh tanda baca koma pada perhitungan ini.

Tabel 4.3 Tokenisasi Data *Training*

No	Tokenisasi
1	karena, @fadlizon, kebanyakan, nyinyir, jadi, otaknya, ga, dipake, mikir
2	@fadlizon, komentar, terus, aja, lu, zon, nyinyiranmu, bikin, eneg
3	@fadlizon, goblok, gak, mikir, lo, ye, makin, lo, nyinyir, gak, pake, otak, kaya, gini, makin, eneg, orang, liat, muke, lo, dasar, tukang, nyinyir
4	pak, @fadlizon, juga, punya, prestasi, yang, membanggakan, kok, makanya, bisa, duduk, di, DPR
5	bangga, sama, bang, @fadlizon, mantap, tidak, takut
6	@fadlizon, walaupun, anda, dihina, terus, sy, tetap, bangga, sama, anda, pak, dengan, prestasi, bapak, mantap

Selanjutnya manualisasi tokenisasi untuk data *testing* ditunjukan pada Tabel 4.4.

Tabel 4.4 Tokenisasi Data *Testing*

No	Tokenisasi
1	yaelah.., mulai, lagi, si, @fadlizon, nyinyir, di, publik, malu-maluin, goblok
2	@fadlizon, sabar, ya, pak.., prestasi, bapak, akan, mengalahkan, omongan, mereka, tidak, usah, khawatir, kami, bangga

4.3.1.2 Filtering

Perhitungan hasil token untuk proses *filtering* ditunjukan pada Tabel 4.5 untuk data *training*.

Tabel 4.5 *Filtering* Data *Training*

No	Filtering
1	nyinyir, otaknya, ga, dipake, mikir
2	nyinyiranmu, eneg
3	goblok, gak, mikir, nyinyir, gak, pake, otak, tukang, nyinyir
4	prestasi, membanggakan
5	bangga, mantap, tidak, takut
6	bangga, prestasi, mantap

Selanjutnya hasil token untuk proses *filtering* untuk data *testing* ditunjukan pada Tabel 4.6.

Tabel 4.6 *Filtering* Data *Testing*

No	Filtering
1	nyinyir, malu-maluin, goblok
2	sabar, prestasi, mengalahkan, omongan, tidak, khawatir, bangga

4.3.1.3 *Stemming*

Proses stemming adalah mencari kata dasar dari tiap hasil token dari proses *filtering*. Manualisasi *stemming* ditunjukan pada Tabel 4.7 untuk data *training*.

Tabel 4.7 *Stemming* Data *Training*

No	Stemming
1	nyinyir, otak, tidak, pakai, pikir
2	nyinyir, eneg
3	goblok, tidak, pikir, nyinyir, tidak, pakai, otak, eneg, tukang, nyinyir
4	prestasi, bangga
5	bangga, mantap, tidak takut
6	bangga, prestasi, mantap

Selanjutnya untuk manualisasi *stemming* data *testing* ditunjukan pada Tabel 4.8.

Tabel 4.8 *Stemming* Data *Testing*

No	Stemming
1	nyinyir, malu, goblok
2	sabar, prestasi, kalah, omong, tidak, khawatir, bangga

4.3.1.4 Term Weighting

Tahap pertama yang dilakukan yaitu mencari *term frequency*. Proses ini dilakukan dengan perhitungan jika *term* berada pada suatu dokumen maka akan ditambahkan nilai 1, jika tidak maka akan bernilai 0. Hasil penghitungan *term frequency* ditunjukan pada Tabel 4.9.

Tabel 4.9 *Term Frequency* Data *Training*

Term	TF					
	DOK1	DOK2	DOK3	DOK4	DOK5	DOK6
Nyinyir	1	1	2	0	0	0
Otak	1	0	1	0	0	0
Tidak	1	0	2	0	1	0
Pakai	1	0	1	0	0	0
Pikir	1	0	1	0	0	0
Eneg	0	1	1	0	0	0
Goblok	0	0	1	0	0	0
Tukang	0	0	1	0	0	0
Prestasi	0	0	0	1	0	1
Bangga	0	0	0	1	1	1
Mantap	0	0	0	0	1	1
Takut	0	0	0	0	1	0

Tahap kedua yaitu menghitung bobot tiap kata dengan rumus *weight term frequency.* Contoh perhitungan *weight term frequency* untuk kata "nyinyir" pada dokumen DOK1 ditunjukan pada Persamaan (4.1), dan hasil perhitungan seluruh dokumen ditunjukan pada Tabel 4.10.

$$W_{tf} = 1 + \log_{10} tf = 1 + \log_{10} 1 = 1 \quad (4.1)$$

Tabel 4.10 *Weight Term Frequency* Data *Training*

Term	W_{tf}					
	DOK1	DOK2	DOK3	DOK4	DOK5	DOK6
Nyinyir	1	1	1.301	0	0	0
Otak	1	0	1	0	0	0
Tidak	1	0	1.301	0	1	0
Pakai	1	0	1	0	0	0
Pikir	1	0	1	0	0	0
Eneg	0	1	1	0	0	0
Goblok	0	0	1	0	0	0
Tukang	0	0	1	0	0	0
Prestasi	0	0	0	1	0	1
Bangga	0	0	0	1	1	1
Mantap	0	0	0	0	1	1
Takut	0	0	0	0	1	0

Tahap ketiga yaitu menghitung *document frequency,* dengan menghitung *term* yang muncul pada tiap dokumen. Proses ini menghitung frekuensi kemunculan kata pada tiap dokumen. Hal ini berguna untuk mengetahui di dokumen mana saja kata tersebut muncul. *Document frequency* ditunjukan pada Tabel 4.11.

Tabel 4.11 *Document Frequency* Data *Training*

Term	DF_t
Nyinyir	3
Otak	2
Tidak	3
Pakai	2
Pikir	2
Eneg	2
Goblok	1
Tukang	1
Prestasi	2
Bangga	3
Mantap	2
Takut	1

Tahap keempat yaitu menghitung *inverse document frequency*. Contoh perhitungan IDF untuk kata "nyinyir" ditunjukan dapa Persamaan (4.2), imana N merupakan jumlah dokumen dan df_t merupakan *document frequency*. Hasil perhitungan seluruh kata ditunjukan pada Tabel 4.12.

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right) = \log_{10}\left(\frac{6}{3}\right) = 0.301 \quad (4.2)$$

Tabel 4.12 *Inverse Document Frequency* Data *Training*

Term	IDF_t
Nyinyir	0.301
Otak	0.477
Tidak	0.301
Pakai	0.477
Pikir	0.477
Eneg	0.477
Goblok	0.778
Tukang	0.778
Prestasi	0.477
Bangga	0.301
Mantap	0.477
Takut	0.778

Tahap terakhir yaitu menghitung *weight term document*. Untuk bobot kata pada tiap dokumen data *training* yang bertujuan untuk mengetahui seberapa penting kata pada tiap dokumen. Contoh perhitungan bobot kata "nyinyir" pada dokumen DOK1 ditunjukan pada Persamaan (4.3), dan hasil perhitungan seluruh kata ditunjukan pada Tabel 4.13.

$$W_{td} = idf_t \times tf = 0.301 \times 1 = 0.301 \quad (4.3)$$

Tabel 4.13 *Weight Term Document* Data *Training*

Term	W_{td}					
	DOK1	DOK2	DOK3	DOK4	DOK5	DOK6
Nyinyir	0.301	0.301	0.391	0	0	0
Otak	0.477	0	0.477	0	0	0
Tidak	0.301	0	0.391	0	1	0
Pakai	0.477	0	0.477	0	0	0
Pikir	0.477	0	0.477	0	0	0
Eneg	0	0.477	0.477	0	0	0
Goblok	0	0	0.778	0	0	0
Tukang	0	0	0.778	0	0	0
Prestasi	0	0	0	0.477	0	0.477
Bangga	0	0	0	0.301	0.301	0.301

Mantap	0	0	0	0	0.477	0.477
Takut	0	0	0	0	0.778	0

Hasil dari *weight term document* akan digunakan untuk klasifikasi pada metode SVM. Hasil dari IDF_t akan disimpan dan akan digunakan pada data *testing* untuk seleksi fitur IG dan klasifikasi SVM.

4.3.2 Information Gain

Tahap pertama dalam proses menghitung nilai *information gain* adalah dengan mencari *term Presence* kata pada tiap dokumen. Contoh perhitungan manual untuk *term Presence*, misalnya jika kata "nyinyir" muncul pada dokumen DOK1 maka akan diberi nilai 1, sedangkan jika tidak maka diberi nilai 0. Hasil penghitungan kemunculan kata ditunjukan pada Tabel 4.14.

Tabel 4.14 *Term Presence* Data *Training*

Term	TP					
	DOK1	**DOK2**	**DOK3**	**DOK4**	**DOK5**	**DOK6**
Nyinyir	1	1	1	0	0	0
Otak	1	0	1	0	0	0
Tidak	1	0	1	0	1	0
Pakai	1	0	1	0	0	0
Pikir	1	0	1	0	0	0
Eneg	0	1	1	0	0	0
Goblok	0	0	1	0	0	0
Tukang	0	0	1	0	0	0
Prestasi	0	0	0	1	0	1

Bangga	0	0	0	1	1	1
Mantap	0	0	0	0	1	1
Takut	0	0	0	0	1	0

Tahap selanjutnya adalah proses menghitung jumlah nilai 1 dan nilai 0 yang ada untuk mengetahui jumlah nilai 1 dan nilai 0. Jumlah nilai 1 dan nilai 0 digunakan untuk perhitungan nilai *Information Gain*. Hasil perhitungan jumlah nilai 1 dan 0 ditunjukan pada Tabel 4.15.

Tabel 4.15 Jumlah Nilai 1 dan 0

Term	Jumlah 1	Jumlah 0
Nyinyir	3	3
Otak	2	4
Tidak	3	3
Pakai	2	4
Pikir	2	4
Eneg	2	4
Goblok	1	5
Tukang	1	5
Prestasi	2	4
Bangga	3	3
Mantap	2	4
Takut	1	5

Tahap selanjutnya yaitu menghitung nilai 1 pada tiap kelas *bully* dan kelas bukan *bully*. Proses menghitung nilai 1 pada masing-masing kelas bertujuan untuk mengetahui termasuk dalam kelas apakah kata tersebut dan seberapa penting kata tersebut. Hasil perhitungan ditunjukan pada Tabel 4.16.

Tabel 4.16 Jumlah Nilai 1 pada tiap Kelas

Term	*Bully*	*No*
Nyinyir	3	0
Otak	2	0
Tidak	2	1
Pakai	2	0
Pikir	2	0
Eneg	2	0
Goblok	1	0

Tukang	1	0
Prestasi	0	2
Bangga	0	3
Mantap	0	2
Takut	0	1

Tahap selanjutnya yaitu menghitung nilai IG pada tiap fitur. Contoh perhitungan kata "nyinyir" ditunjukan pada Persamaan (4.4), dimana c merupakan kelas, C_i merupakan dokumen, $P(C_i)$ merupakan peluang kelas dokumen, $P(t)$ merupakan peluang kata dari seluruh jumlah kata, $P(C_i|t)$ merupakan peluang kata muncul pada dokumen. Hasil perhitungan seluruh kata ditunjukan pada Tabel 4.17.

$$IG(t) = -\sum_{i=1}^{|C|} P(C_i)\log P(C_i) + P(t)\sum_{i=1}^{|C|} P(C_i|t)\log P(C_i|t) + P(\bar{t})\sum_{i=1}^{|C|} P(C_i|\bar{t})\log P(C_i|\bar{t}) = -\left(\left(\frac{3}{6}\log\frac{3}{6}\right)+\left(\frac{3}{6}\log\frac{3}{6}\right)\right) + \frac{3}{24}\left(\left(\frac{3}{3}\log\frac{3}{3}\right)+\left(\frac{0}{3}\log\frac{0}{3}\right)\right)+\frac{3}{24}\left(\left(\frac{0}{3}\log\frac{0}{3}\right)+\left(\frac{3}{3}\log\frac{3}{3}\right)\right) = 0.301 \quad (4.4)$$

Tabel 4.17 *Information Gain*

Term	*IG*
Nyinyir	0.301
Otak	0.264
Tidak	0.129
Pakai	0.264
Pikir	0.264
Eneg	0.264
Goblok	0.269
Tukang	0.269
Prestasi	0.264
Bangga	0.301
Mantap	0.264
Takut	0.269

Setelah mendapatkan nilai IG tiap kata, selanjutnya melakukan pengurutan untuk mengetahui nilai IG terbesar hingga terkecil. Hal ini berguna untuk menyeleksi fitur-fitur yang bernilai tinggi. *Threshold* yang digunakan pada penelitian ini ada tiga yaitu fitur sebanyak 25%, 50%, 75% dan 100%. Hal ini bertujuan untuk mengetahui pengaruh penggunaan seleksi fitur IG. Hasil pengurutan dan penetapan *threshold* ditunjukan pada Tabel 4.18.

Tabel 4.18 Hasil Seleksi Fitur

Threshold			
25%	**50%**	**75%**	**100%**
Nyinyir	Nyinyir	Nyinyir	Nyinyir
Bangga	Bangga	Bangga	Bangga
Goblok	Goblok	Goblok	Goblok
	Tukang	Tukang	Tukang
	Takut	Takut	Takut
	Otak	Otak	Otak
		Pakai	Pakai
		Piker	Piker
		Eneg	Eneg
			Prestasi
			Mantap
			Tidak

4.3.3 Support Vector Machine

Nilai masukan pada proses ini adalah nilai dari fitur hasil *text preprocessing*. Proses pada metode SVM mempunyai beberapa parameter yaitu: nilai $c = 1, d = 2$, $\alpha_i = 0$, $\lambda = 0.5$, $\gamma = 0.001$, $\varepsilon = 0.0001$, $IterMax = 2$. Tahap perhitungan untuk metode SVM yaitu:

1. Menghitung Kernel Polynomial d
2. Menghitung *Matriks Hessian*
3. Menghitung *Sequential Training* SVM
   a. Menghitung nilai E_i
   b. Menghitung nilai $\delta\alpha$
   c. Menghitung nilai α_i

4.3.3.2 Kernel Polynomial d

Proses perhitungan Kernel Polynomial dilakukan setelah data *training* diinputkan, data *training* didapatkan dari hasil seleksi fitur IG. Dimana dalam perhitungan ini jumlah *threshold* yang digunakan yaitu 100%. Data *training* SVM ditunjukan pada Tabel 4.19.

Tabel 4.19 Input Data Training SVM

Term	W_{td}					
	DOK1	**DOK2**	**DOK3**	**DOK4**	**DOK5**	**DOK6**

Nyinyir	0.301	0.301	0.391	0	0	0
Otak	0.477	0	0.477	0	0	0
Tidak	0.301	0	0.391	0	1	0
Pakai	0.477	0	0.477	0	0	0
Pikir	0.477	0	0.477	0	0	0
Eneg	0	0.477	0.477	0	0	0
Goblok	0	0	0.778	0	0	0
Tukang	0	0	0.778	0	0	0
Prestasi	0	0	0	0.477	0	0.477
Bangga	0	0	0	0.301	0.301	0.301
Mantap	0	0	0	0	0.477	0.477
Takut	0	0	0	0	0.778	0

Contoh perhitungan untuk dokumen DOK1 ditunjukan pada Persamaan (4.5). dimana d merupakan ordo dan c merupakan *complexity*. Dan hasil perhitungan seluruh dokumen ditunjukan pada Tabel 4.20.

$$K(x_i, x_j) = (x_i \cdot x_j + c)^d = ((0.301 \cdot 0.301) + (0.477 \cdot 0.477) + (0.301 \cdot 0.301) + (0.477 \cdot 0.477) + (0.477 \cdot 0.477) + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1)^2 = 3.475138 \quad (4.5)$$

Tabel 4.20 Hasil Kernel Polynomial Data Training

Polynomial	DOK1	DOK2	DOK3	DOK4	DOK5	DOK6	KELAS
DOK1	3.475138	1.18945	3.681526	1	1.18945	1	1
DOK2	1.18945	1.737819	1.810485	1	1	1	1
DOK3	3.681526	1.810485	11.7539	1	1.249696	1	1
DOK4	1	1	1	1.737819	1.18945	1.737819	-1
DOK5	1.18945	1	1.249696	1.18945	4.057816	1.737819	-1
DOK6	1	1	1	1.737819	1.737819	2.389833	-1
KELAS	1	1	1	-1	-1	-1	

4.3.3.3 *Matriks Hessian*

Perhiitungan *Matriks Hessian* untuk dokumen DOK1 ditunjukan pada Persamaan (4.6). Dimana D_{ij} merupakan *Matriks Hessian*, $y_i y_j$ merupakan kelas masing-masing dokumen. Hasil perhitungan *Matriks Hessian* ditunjukan pada Tabel 4.21.

$$D_{ij} = y_i y_j \left(K(x_i \cdot x_j) + \lambda^2\right) = 1 \times 1(3.475138 + 0.5^2) = 3.725 \quad (4.6)$$

Tabel 4.21 Hasil *Matriks Hessian* Data *Training*

Hessian	DOK1	DOK2	DOK3	DOK4	DOK5	DOK6
DOK1	3.725138	1.43945	3.931526	-1.25	-1.43945	-1.25
DOK2	1.43945	1.987819	2.060485	-1.25	-1.25	-1.25
DOK3	3.931526	2.060485	12.0039	-1.25	-1.4997	-1.25
DOK4	-1.25	-1.25	-1.25	1.987819	1.43945	1.987819
DOK5	-1.43945	-1.25	-1.4997	1.43945	4.307816	1.987819
DOK6	-1.25	-1.25	-1.25	1.987819	1.987819	2.639833

4.3.3.4 *Sequential Training* SVM

Perhitungan sequential training SVM. Dalam perhitungan training jumlah iterasi yang digunakan sebanyak 2 iterasi, diamana nilai γ didapatkan dari hasil *konstanta*/*maxHessian*. Contoh perhitungan dokumen DOK1 untuk nilai E_i, $\delta\alpha$, dan α_i ditunjukan pada Persamaan (4.7), (4.8), dan (4.9). Hasil perhitungan ditunjukan pada Tabel 4.22.

$$E_i = \sum_i^N \alpha_i D_{ij} = 0 \quad (4.7)$$

$$\delta\alpha = \min\left\{\max(\gamma(1 - E_i), -\alpha_i)c - \alpha_i\right\} = \min\left\{\max(0.000254(1 - 0), 0)1 - 0\right\} = 0.000254 \quad (4.8)$$

$$\alpha_i = \alpha_i + \delta\alpha_i = 0 + 0.000254 = 0.000254 \quad (4.9)$$

Tabel 4.22 *Sequential Training* SVM

Iterasi 0	DOK1	DOK2	DOK3	DOK4	DOK5	DOK6
$\boldsymbol{E_i}$	0	0	0	0	0	0
$\boldsymbol{\delta\alpha}$	0,000254	0,000254	0,000254	0,000254	0,000254	0,000254
$\boldsymbol{\alpha_i}$	0,000254	0,000254	0,000254	0,000254	0,000254	0,000254
Iterasi 1						
$\boldsymbol{E_i}$	0,001312	0,000442	0,00356	0,000424	0,000902	0,000729
$\boldsymbol{\delta\alpha}$	0,000254	0,000254	0,000253	0,000254	0,000254	0,000254
$\boldsymbol{\alpha_i}$	0,000508	0,000509	0,000508	0,000509	0,000508	0,000509
Iterasi 2						
$\boldsymbol{E_i}$	0,002619	0,000882	0,007108	0,000848	0,001804	0,001458
$\boldsymbol{\delta\alpha}$	0,000254	0,000254	0,000253	0,000254	0,000254	0,000254
$\boldsymbol{\alpha_i}$	0,000762	0,000763	0,00076	0,000763	0,000762	0,000763

4.3.3.5 Menentukan Nilai *Support Vector*

Proses selanjutnya yaitu mencari nilai *Support Vector* Positif dan *Support Vector* Negatif. Nilai yang dihasilkan dari iterasi terakhir menunjukan enam data *training* memiliki nilai lebih dari nol. Hasil *support vector* data *training* ditunjukan pada Tabel 4.23.

Tabel 4.23 Support Vector Data Training

SV	α_i	Kelas
DOK1	0,000762	1
DOK2	0,000763	1
DOK3	0,00076	1
DOK4	0,000763	-1
DOK5	0,000762	-1
DOK6	0,000763	-1

Setelah mendapatkan nilai positif dan negatif dokumen pada tiap kelas, selanjutnya mencari nilai alpha maksimal pada kelas positif dan nilai alpha maksimal pada kelas negatif, yang ditunjukan pada Table 4.24.

Tabel 4.24 Nilai Max SV Data Training

	Max X+	Max X-
α_i	0,000763	0,000763
SV	2	4

4.3.3.6 Menghitung Bobot Data dan Bias

Proses selanjutnya yaitu menghitung bobot data *training* yang memiliki nilai alpha tertinggi pada kelas positif dan yang tertinggi pada kelas negatif. Bobot kelas negatif merupakan Kernel dari data *training* yang memiliki kelas negatif, sedangkan bobot kelas positif menggunakan Kernel dari data *training* yang memiliki kelas positif. Kemudian menghitung nilai bias SVM. Dalam proses perhitungan nilai bias dibutuhkan total jumlah bobot keals positif dan negatif. Rumus perhitungan bias ditunjukan pada Persamaan (4.10) dan hasil perhitungan kelas positif dan negatif serta nilai bias ditujunkan pada Tabel 4.25.

$$b = -\frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x^+) + \sum_{i=0}^{n} \alpha_i y_i K(x_i, x^-)\right) \quad (4.10)$$

Tabel 4.25 Nilai Bias Data Training

Nilai b	$K(x_i, x^+)$	$K(x_i, x^-)$	$\alpha_i y_i K(x_i, x^+)$	$\alpha_i y_i K(x_i, x^-)$
DOK1	1.18945	1	0,000906	0,000762
DOK2	1.737819	1	0,001325	0,000763

DOK3	1.810485	1	0,001377	0,00076
DOK4	1	1.737819	-0,00076	-0,00133
DOK5	1	1.18945	-0,00076	-0,00091
DOK6	1	1.737819	-0,00076	-0,00133
$\sum_{i=1}^{n} \alpha_i y_i K(x_i, x^+) + \sum_{i=0}^{n} \alpha_i y_i K(x_i, x^-)$			0,001321	-0,00127
$\boldsymbol{b}$			-0.0000243	

4.3.3.7 *Testing* SVM

Pada perhitungan untuk *testing* SVM menggunakan data *testing* hasil *stemming* yang ditunjukan pada Tabel 4.8. *Term frequency* data *testing* dihitung berdasarkan *list term frequency* dari data *training*. Hasil *term frequency* data *testing* ditunjukan pada Tabel 4.26.

Tabel 4.26 Term Frequency Data Testing

Term	DOK7	DOK8
Nyinyir	1	0
Otak	0	0
Tidak	0	1
Pakai	0	0
Pikir	0	0
Eneg	0	0
Goblok	1	0
Tukang	0	0
Prestasi	0	1
Bangga	0	1
Mantap	0	0
Takut	0	0

Proses selanjutnya yaitu menghitung nilai IDF dari TF data *testing* berdasarkan data *training*. Hasil IDF kemudian menjadi input untuk testing SVM. Hasil perkalian TF data *testing* dengan IDF data *training* ditunjukan pada Tabel 4.27.

Tabel 4.27 Input Data Testing SVM

Term	DOK7	DOK8
Nyinyir	0.301	0
Otak	0	0

Tidak	0	0.301
Pakai	0	0
Pikir	0	0
Eneg	0	0
Goblok	0.778	0
Tukang	0	0
Prestasi	0	0.477
Bangga	0	0.301
Mantap	0	0
Takut	0	0

Proses testing SVM dimulai dengan menghitung Kernel Polynomial dokumen data *testing* dengan dokumen data *training*. Contoh perhitungan Kernel Polynomial untuk dokumen DOK1 dan DOK7 ditunjukan pada Persamaan (4.11), dan hasil perhitungan Kernel Polynomial ditunjukan pada Tabel 4.28.

$$K(x_i, x_j) = (x_i \cdot x_j + c)^d = ((0.301 \cdot 0.301) + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1)^2 = 1.18945 \quad (4.11)$$

Tabel 4.28 Hasil Kernel Polynomial Data Testing

Polynomial	DOK7	DOK8
DOK1	1.18945	1.18945
DOK2	1.18945	1
DOK3	2.970168	1.249696
DOK4	1	1.737819
DOK5	1	1.395323
DOK6	1	1.737819

Proses selanjutnya yaitu menghitung nilai kelas positif dan negatif dengan melakukan perhitungan alpha baru dikali kelas dan dikali hasil Kernel Polynomial. Contoh perhitungan untuk dokumen DOK1 dan DOK7 ditunjukan pada persamaan (4.12), dan untuk perhitungan $H(x)$ ditunjukan pada persamaan (4.13). Hasil perhitungan kelas positif dan negatif ditunjukan pada Table 4.29.

$$\alpha_i y_i K(x_i, x_j) = 0.002985 \cdot 1 \cdot 1.18945 = 0.00355 \quad (4.12)$$

$$H(x) = \sum_{i=1}^{n} \left(\alpha_i y_i K(x_i, x_j)\right)(b) = (0.00355 + 0.003562 + 0.008786 - 0.00299 - 0.00299 - 0.00299) - 0.0000565 = 0.006866 \quad (4.13)$$

Tabel 4.29 Hasil Data Testing

$\alpha_i y_i K(x_i, x_j)$	DOK7	DOK8
DOK1	0,000906	0,000906
DOK2	0,000907	0,000763
DOK3	0,002258	0,00095
DOK4	-0,00076	-0,00133
DOK5	-0,00076	-0,00106
DOK6	-0,00076	-0,00133
$H(x)$	0,00176	-0,00112

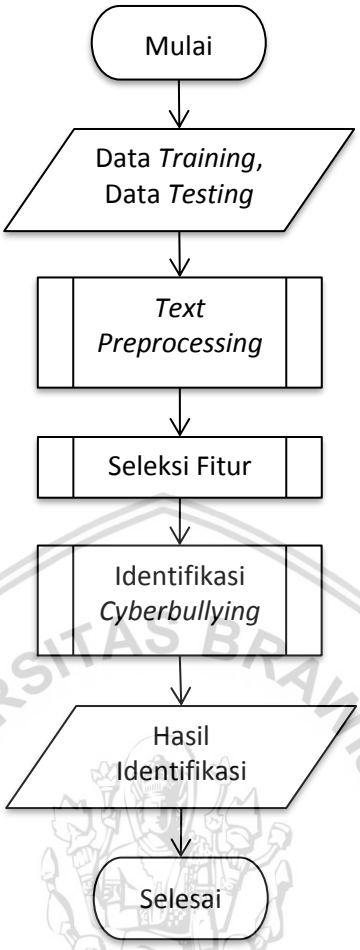
Kemudian hasil dari perhitungan kelas positif dan kelas negatif dilakukan pengklasifikasian data *testing*. Kelas positif menunjukan bahwa dokumen merupakan *cyberbullying*, sedangan kelas negatif menunjukan kelas bukan *cyberbullying*. Hasil klasifikasi ditunjukan pada Table 4.30.

Tabel 4.30 Hasil Klasfikasi SVM Data Testing

Klasifikasi	$H(x)$	Sistem	Manual	KET
DOK7	0,00176	1	1	TRUE
DOK8	-0,00112	-1	-1	TRUE

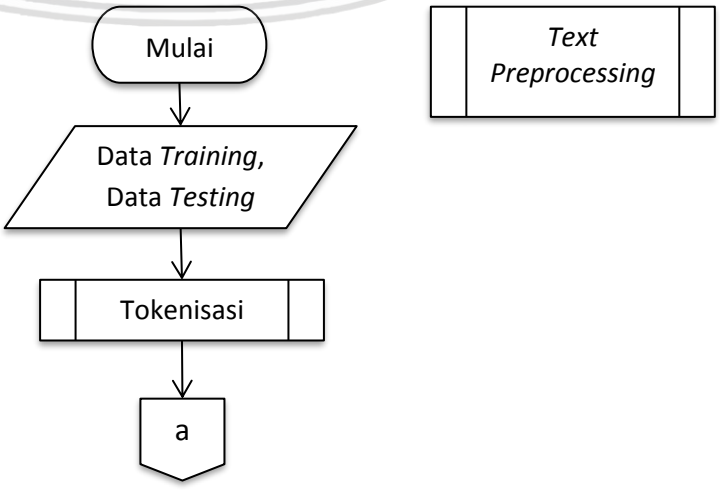
4.4 Diagram Alir Sistem

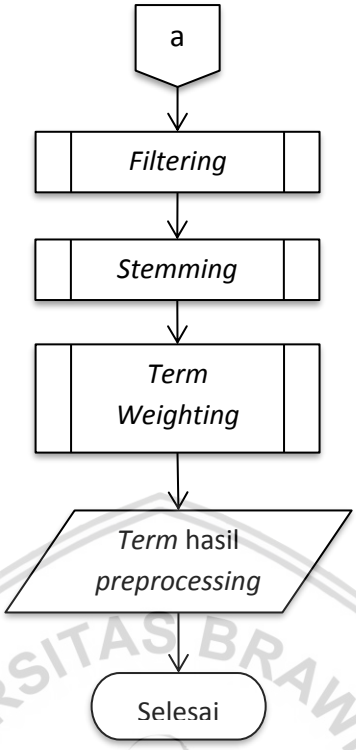
Sistem pada penelitian ini terdapat empat proses yaitu *text preprocessing*, TF-IDF, seleksi fitur *Information Gain*, dan klasifikasi menggunakan metode *Support Vector Machine*. Alur kerja sistem ditunjukan oleh diagram alir pada Gambar 4.1.



Gambar 4.1 Diagram Alir Sistem

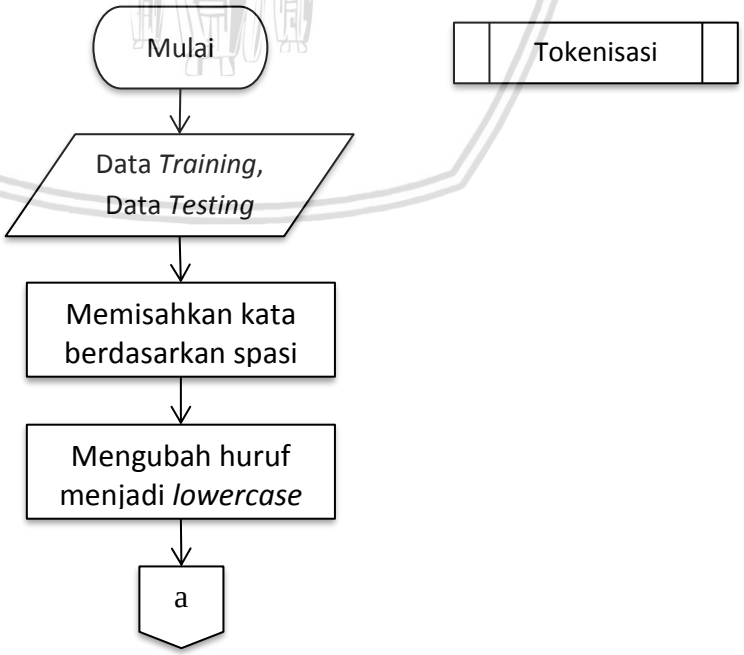
Sistem beke rja mulai dari memproses dokumen *tweet* dengan *text preprocessing*. Kemudian fitur hasil dari pemrosesan teks tersebut akan diseleksi menggunakan *Information Gain* dengan batasan yang telah ditentukan. Setelah seleksi fitur kemudian melakukan identifikasi *tweet bullying* dengan menggunakan metode *Support Vector Machine* (SVM).

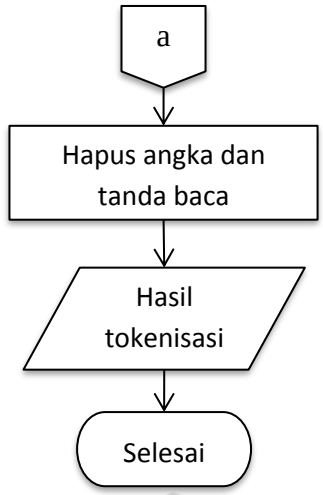




Gambar 4.2 *Text Preprocessing*

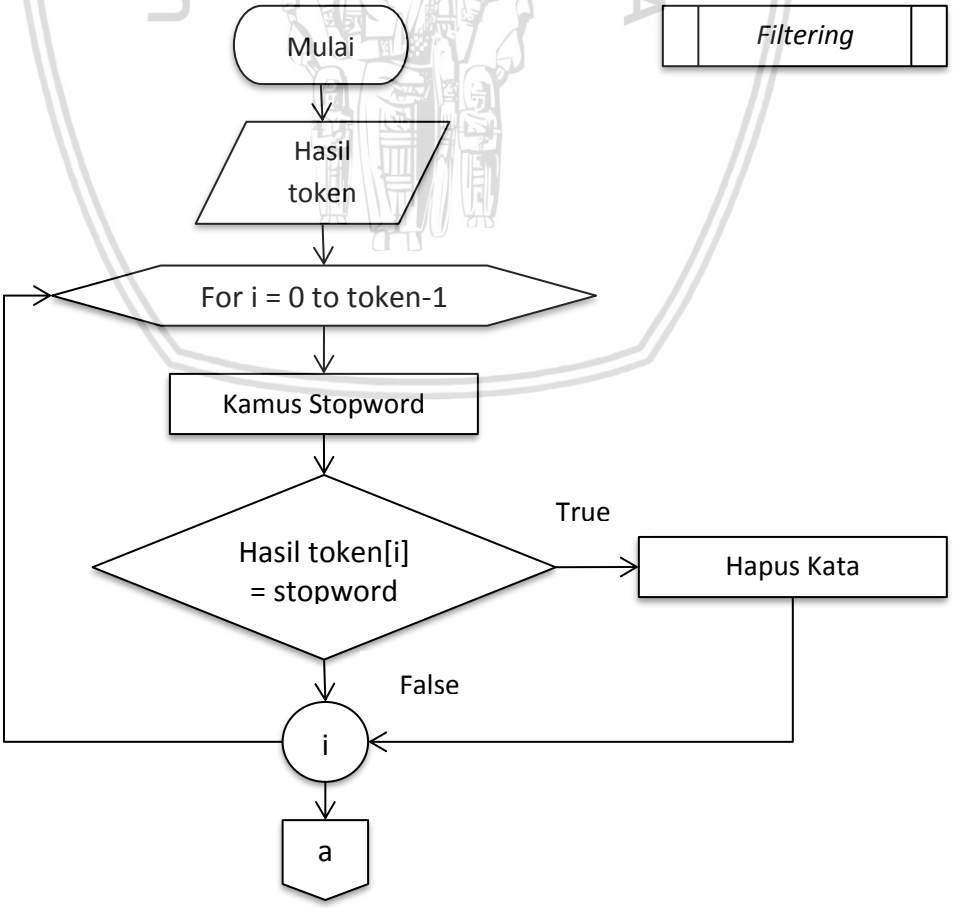
Alur kerja proses *Text Preprocessing* yang ditunjukan pada Gambar 4.2, Proses *text preprocessing* dimulai dari *input* data *training* dan data *testing*, kemudian data masuk ke proses Tokenisasi, *Filtering*, *Stemming* dan proses perhitungan *Term Weighting*. *Output* dari proses ini adalah *term* hasil *preprocessing*.

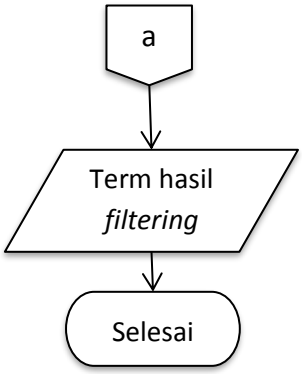




Gambar 4.3 Tokenisasi

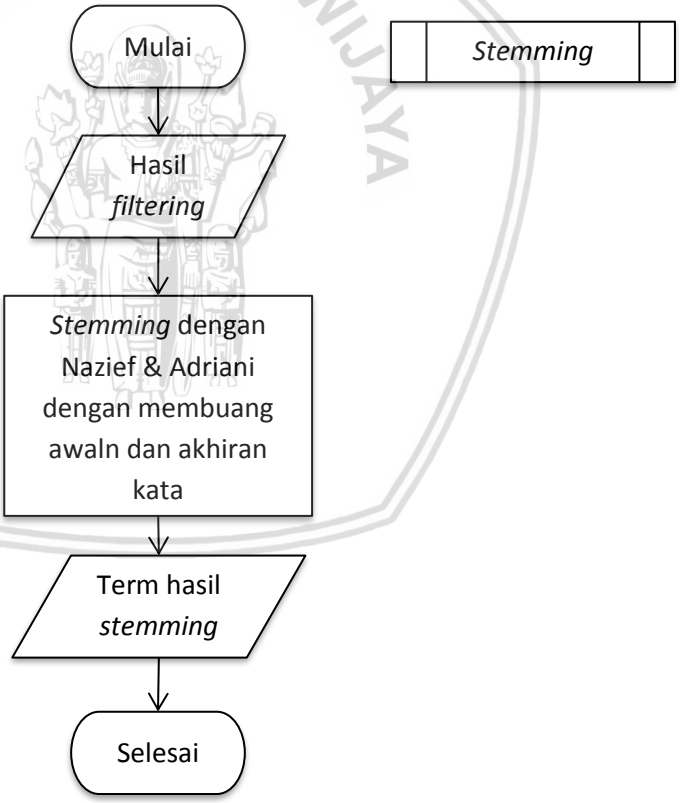
Tokenisasi memisah kata pada tiap dokumen berdasarkan spasi. Proses Tokenisasi yang ditunjukan pada diagram alir Gambar 4.3, dimana proses dimulai dengan *input* data *training* dan *testing* kemudian masuk ke proses memisahkan kata berdasarkan spasi, setelah itu menyelaraskan tiap kata menjadi huruf kecil atau biasa dinamakan *Case Folding* dan proses terakhir menghilangkan angka dan tanda baca. Output dari proses tokenisasi adalah token yang kemudian akan digunakan pada proses selanjutnya.





Gambar 4.4 Filtering

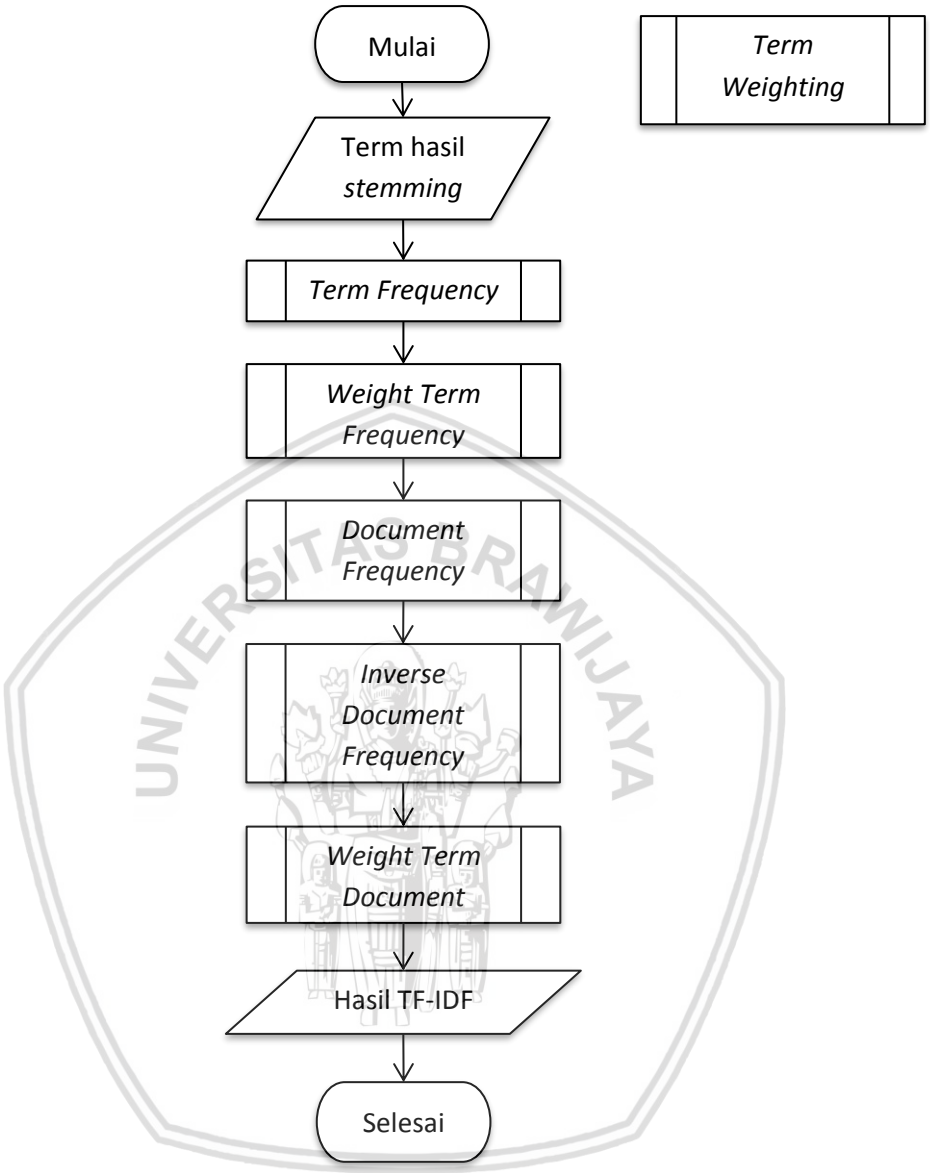
Proses *Filtering* yang ditunjukan pada diagram alir Gambar 4.4 merupakan proses untuk membuang kata yang tidak penting dalam dokumen, dan pemilihan kata menggunakan *stopword*. *Input* dari *filtering* adalah token hasil dari tokenisasi, kemudian masuk ke perulangan untuk mengecek token dengan menggunakan kamus *stopword*. Jika token saat in sama dengan *stopword*, maka token akan dihapus, perulangan dilakukan sebanyak jumlah token.



Gambar 4.5 Stemming

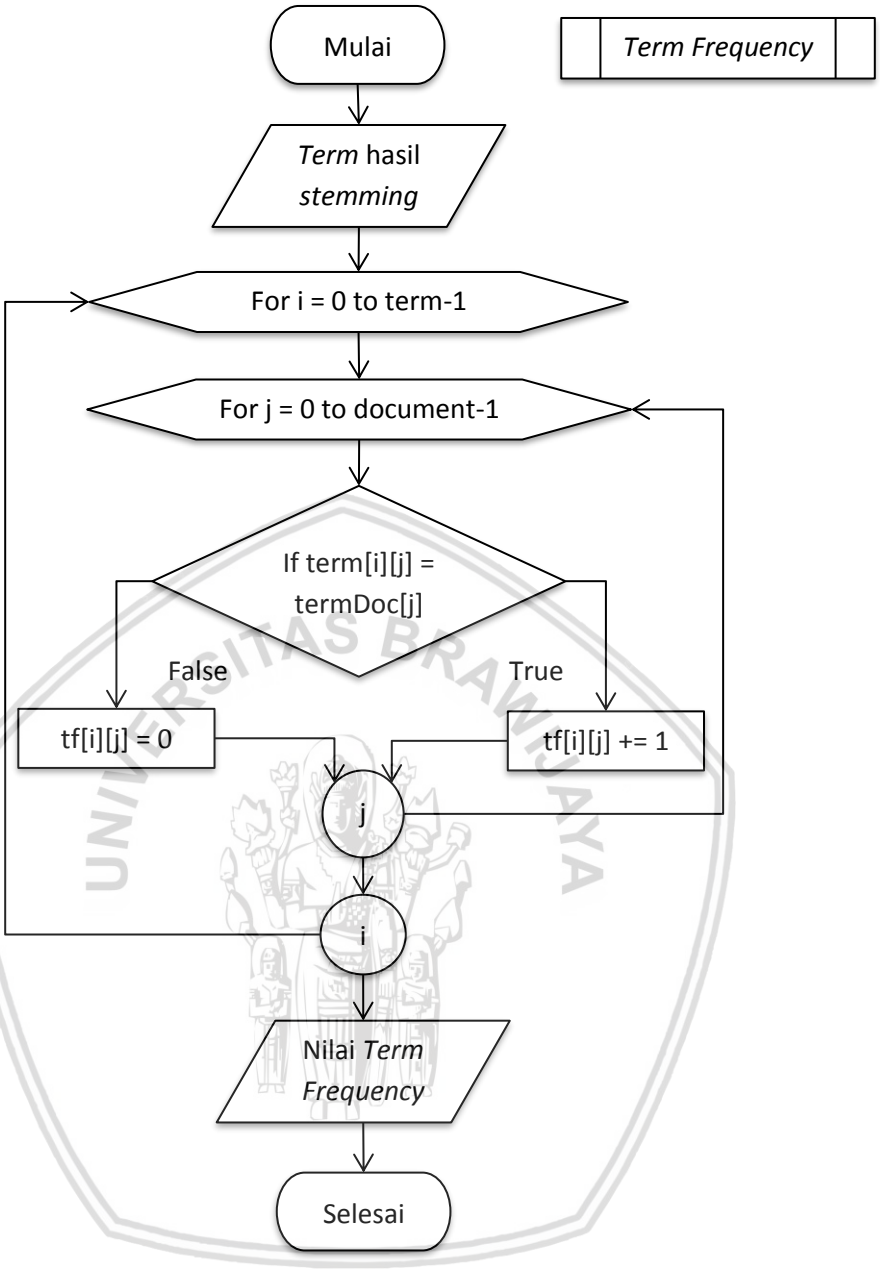
Alur kerja *stemming* ditunjukan pada diagram alir Gambar 4.5. *Stemming* dilakukan untuk mencari kata dasar. Proses *stemming* dimulai dari *input* hasil *fitering*. Kemudian *stemming* pada pemrograman dilakukan dengan menggunakan

stemming Nazief & Adriani. Keluaran dari *stemming* adalah kata dasar dari kata hasil *filtering*.



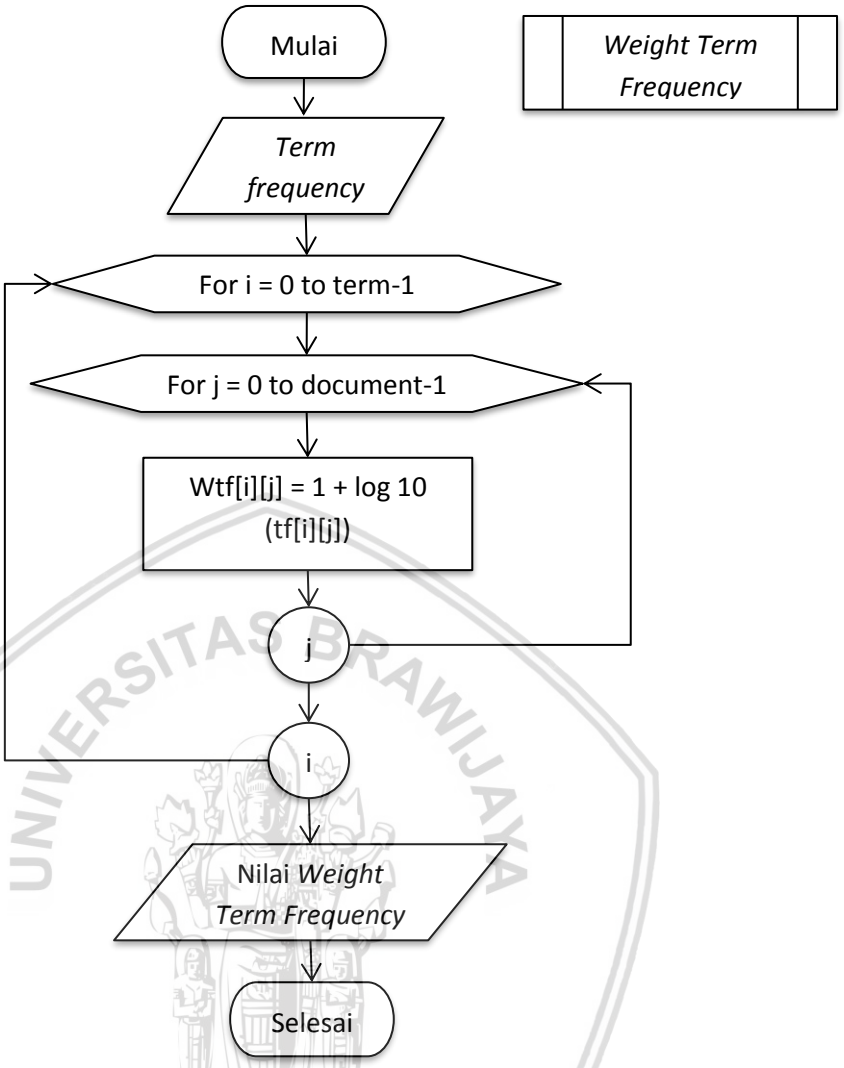
Gambar 4.6 *Term Weighting*

Proses selanjutnya yaitu menghitung *term weighting*. Alur kerja perhitungan *term weighting* ditunjukan pada diagram alir Gambar 4.6. Dalam proses *term weighting* dimulai dengan melakukan perhitungan *term frequency,* kemudian melakukan perhitungan *weight term frequency*. Setelah proses perhitungan masing-masing *term*, selanjutnya menghitung nilai *document frequency*. Proses selanjutnya yaitu menghitung bobot masing-masing *term* pada dokumen dengan *weight term document*. Setelah perhitungan selesai maka didapatkan hasil perhitungan TF-IDF yang selanjutnya digunakan untuk seleksi fitur.



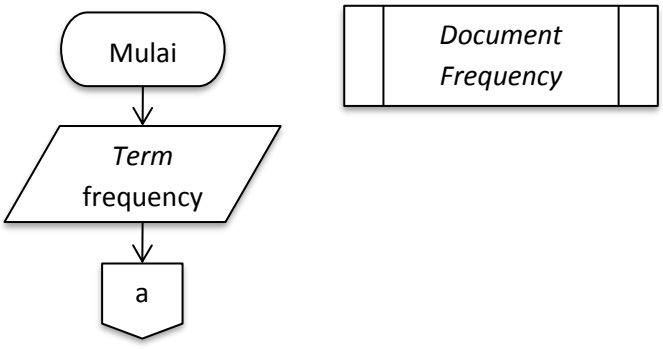
Gambar 4.7 Diagram Alir *Term Frequency*

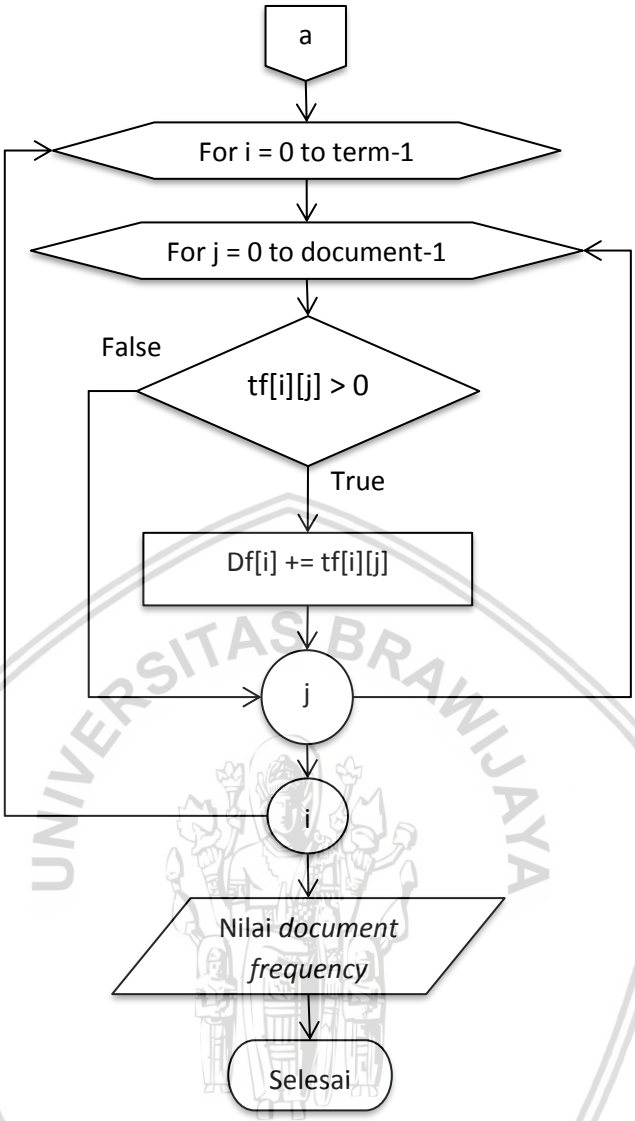
Tahap pertama dalam *term weighting* adalah menghitung *term frequency*. Diagram alir proses *term frequency* ditunjukan pada Gambar 4.7. proses *term frequency* merupakan proses menghitung kemunculan *term* pada suatu dokumen. *Input* proses *term frequency* adalah token hasil *stemming*. Kemudian masuk di proses perulangan dengan mengecek kondisi jika *term* muncul disuatu dokumen maka nilai kemunculan akan ditambahkan 1, jika tidak maka bernilai 0. Setelah perulangan selesai maka keluaran dari proses ini adalah nilai *term frequency*.



Gambar 4.8 Diagram Alir *Weight Term Frequency*

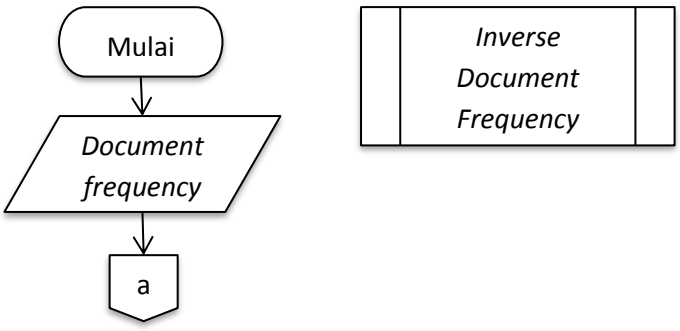
Proses selanjutnya yaitu menghitung *weight term frequency* yang ditunjukan pada diagram alir Gambar 4.8. Proses dimulai dengan input nilai *term frequency*. Kemudian masuk ke perulangan dan proses menghitung *weight term frequency*. Setelah perulangan selesai maka didapatkan keluaran hasil nilai *term weight frequency*.

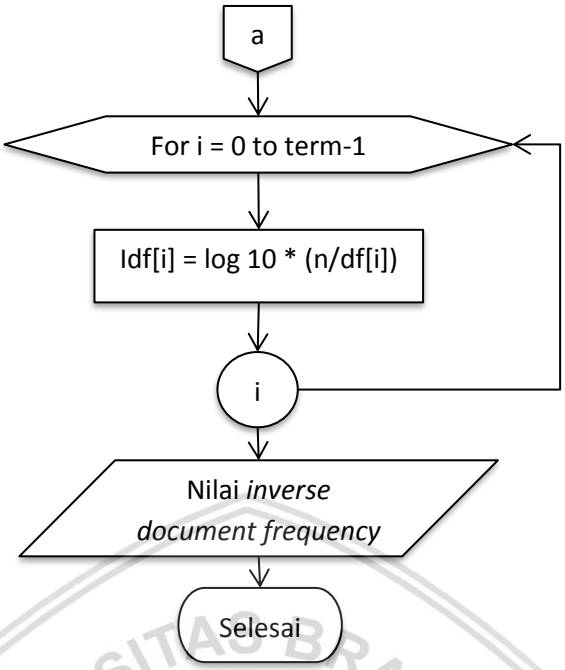




Gambar 4.9 Diagram Alir *Document Frequency*

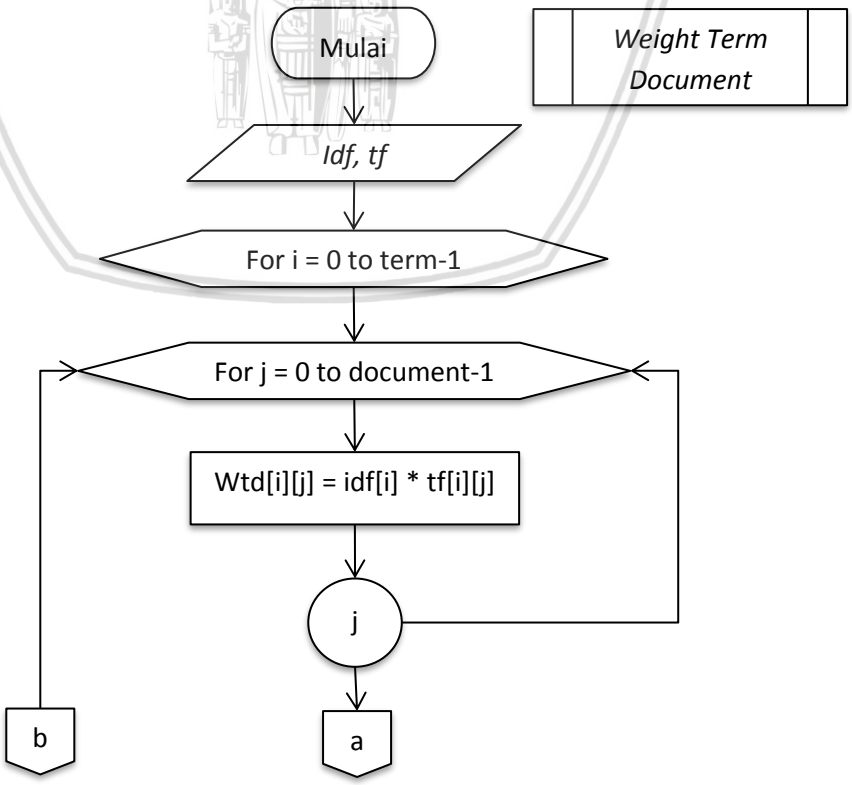
Proses selanjutnya adalah menghitunga *document frequency*. Diagram alir perhitungan *document frequency* ditunjukan pada Gambar 4.9. Proses dimulai dengan *input term* hasil *stemming*. Kemudian masuk ke perulangan dan proses menghitung kemunculan *term* pada dokumen. Setelah perulangan selesai didapatkan keluaran nilai *document frequency*.

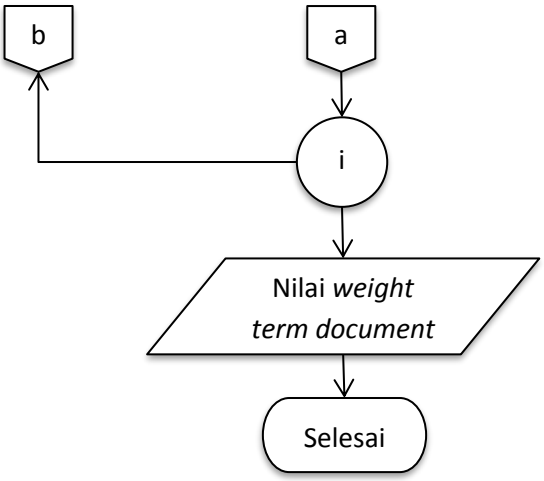




Gambar 4.10 Diagram Alir *Inverse Document Frequency*

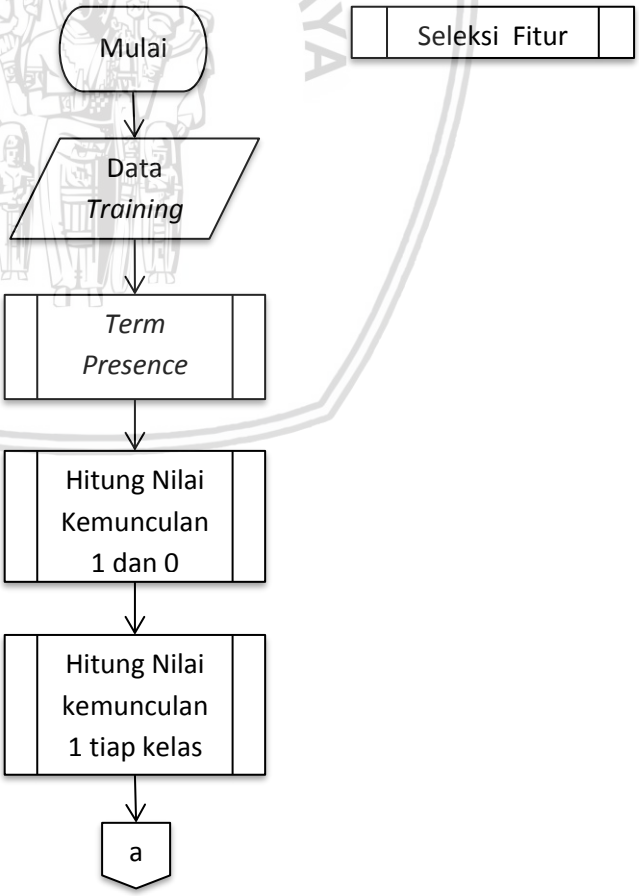
Pada tahap selanjutnya menghitung *inverse* dari *document frequency* yang ditunjukan pada diagram alir Gambar 4.10. Proses dimulai dari *input* nilai hasil *document frequency*, kemudian masuk ke perulangan dan proses menghitung *inverse document frequency*. Keluaran dari proses ini adalah nilai *inverse document frequency*.

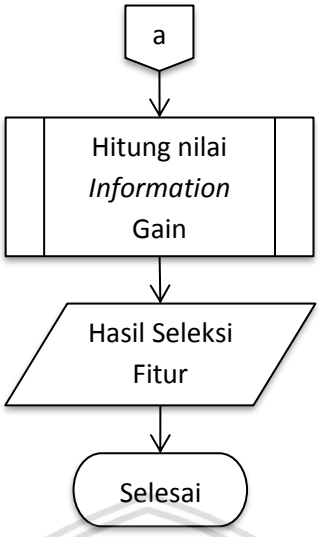




Gambar 4.11 Diagram Alir *Weight Term Document*

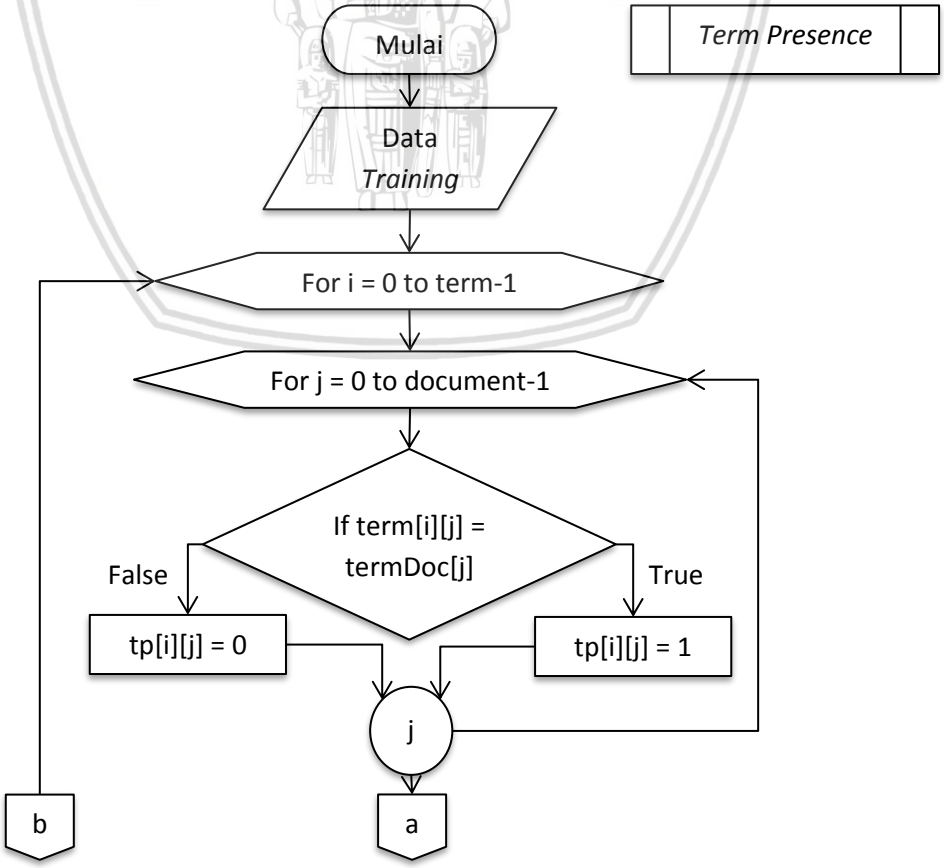
Tahap terakhir dalam *term weighting* yaitu proses *weight term document* yang ditunjukan pada diagram alir Gambar 4.11. Proses ini bertujuan untuk mengetahui seberapa penting kata dalam suatu dokumen. Input dari proses ini adalah nilai hasil *inverse document frequency* dan *term frequency*. Kemudian masuk ke perulangan dan proses perhitungan *weight term document*. Setelah perulangan selesai didapatkan keluaran nilai *weight term document*.

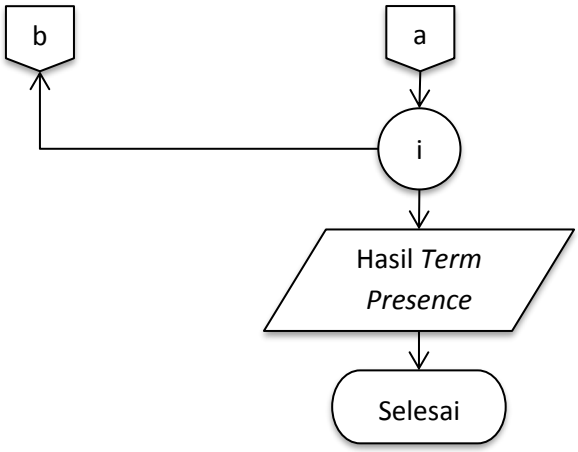




Gambar 4.12 Seleksi Fitur *Information Gain*

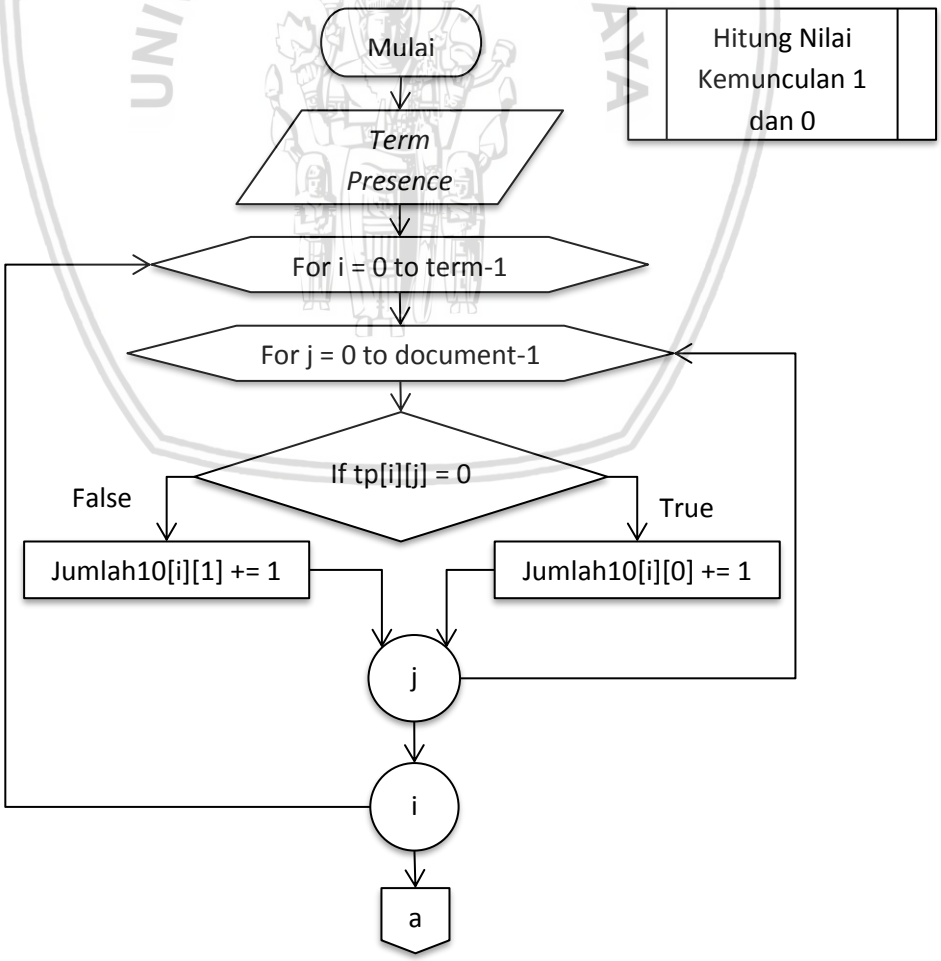
Proses kedua dalam sistem adalah seleksi fitur yang ditunjukan pada diagram alir Gambar 4.12. *Input* untuk seleksi fitur IG adalah data *training,* kemudian masuk ke proses menghitung kemunculan fitur atau term Presence pada dokumen. Kemudian menghitung nilai kemunculan 1 dan 0 hasil term Presence, setelah itu menghitung nilai kemunculan 1. Dan proses terakhir adalah menghitung nilai IG tiap fitur, serta mengurutkan fitur dengan nilai IG tinggi ke rendah.

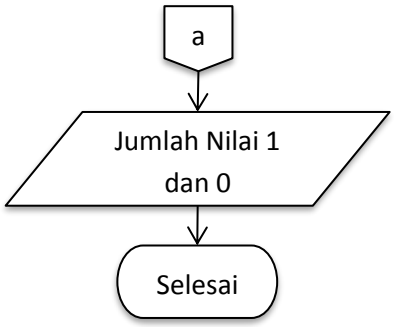




Gambar 4.13 Diagram Alir *Term Presence*

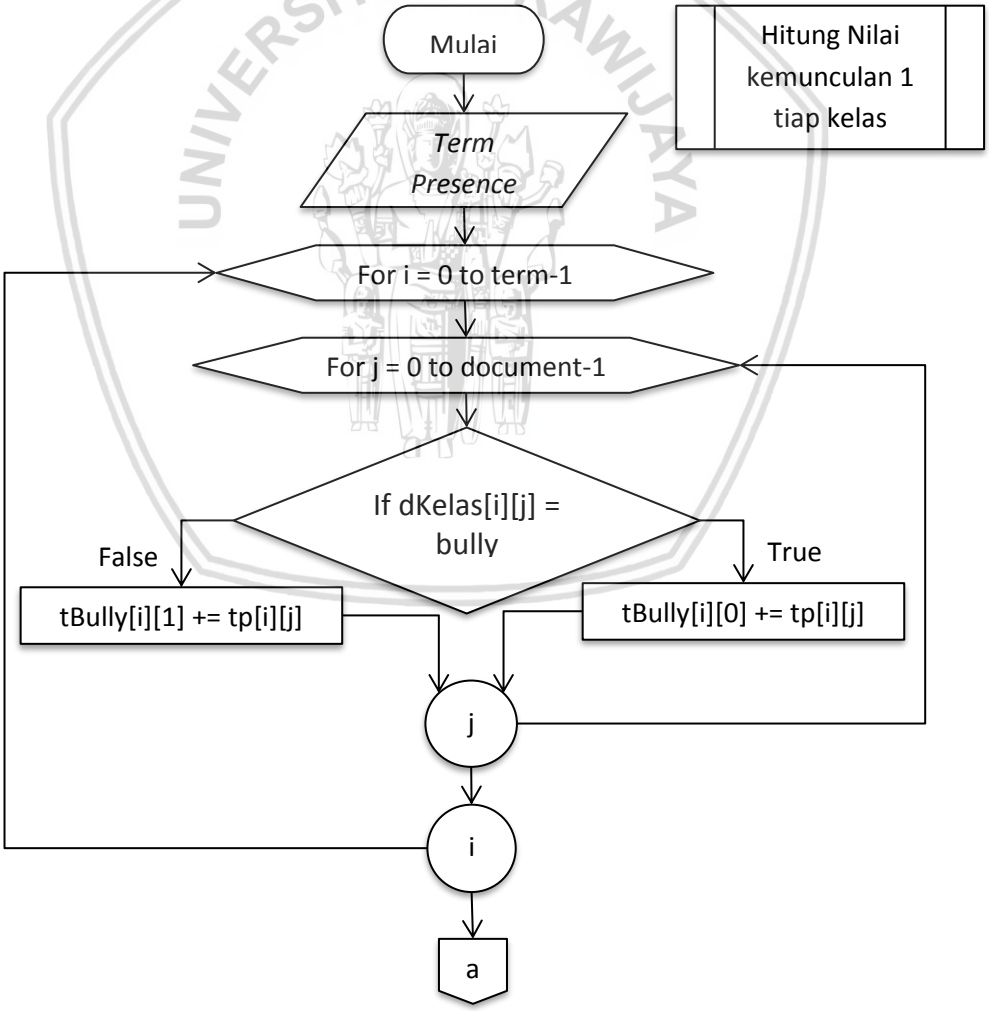
Tahap pertama dalam seleksi fitur *information gain* adalah proses *term Presence* yang ditujukan pada diagram alir Gambar 4.13. *Input* dari perhitungan *term Presence* adalah data *training term* hasil *stemming*. Kemudian masuk ke perulangan dan kondisi jika *term* muncul di dokumen maka variabel *tp* akan bernilai 1, dan jika tidak bernilai 0. Setelah perulangan selesai didapatkan hasil nilai *term Presence*.

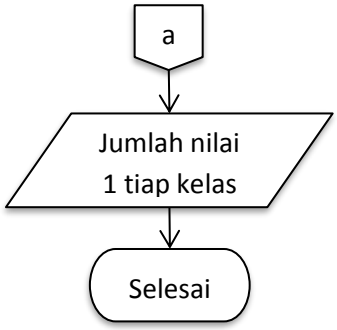




Gambar 4.14 Diagram Alir Jumlah Kemunculan Nilai 1 dan 0

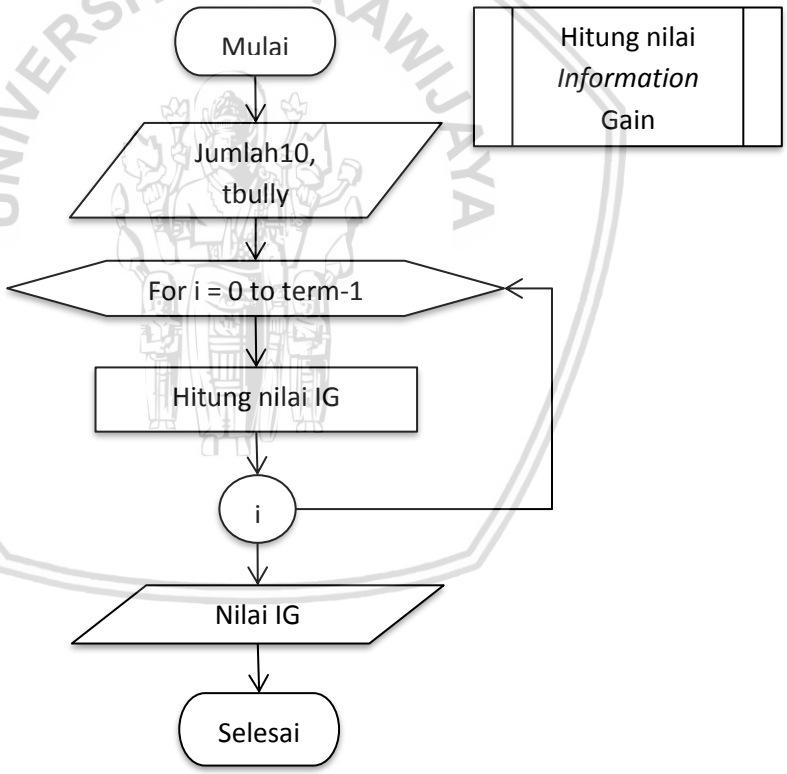
Tahap kedua adalah menghitung jumlah kemunculan nial 1 dan 0 hasil *term Presence*. Alur kerja perhitungan kemunculan nilai 1 dan 0 ditunjukan pada diagram alir Gambar 4.14. *Input* dari proses ini adalah hasil dari perhitungan *term Presence*. Kemudian masuk ke proses perulangan dan menghitung nilai 1 dan 0 yang muncul. Keluaran dari proses ini adalah jumlah nilai 1 dan 0 yang terdapat pada *term Presence*.





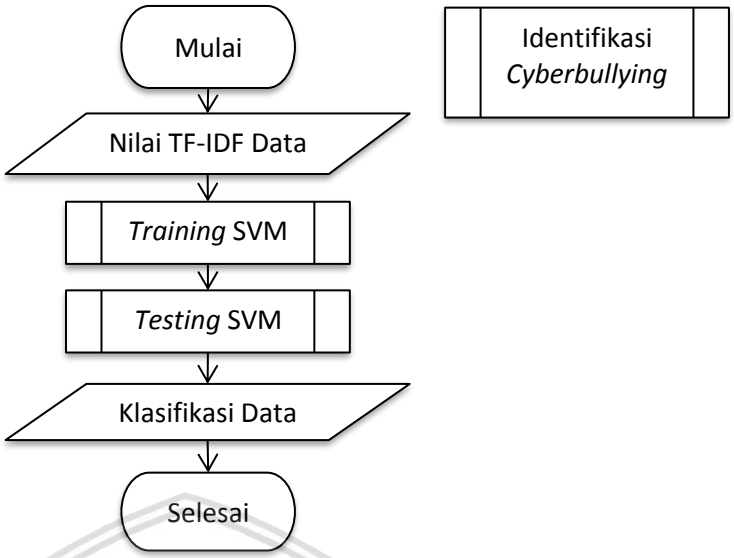
Gambar 4.15 Diagram Alir Jumlah Nilai 1 tiap Kelas

Tahap selanjutnya menghitung nilai 1 yang muncul pada tiap kelas. Alur kerja perhitungan ditunjukan pada diagram alir Gambar 4.15. *Input* dari proses ini adalah hasil dari perhitungan *term Presence*, kemudian masuk ke perulangan dengan kondisi jika dokumen masuk ke kelas *bully* maka menghitung nilai *term Presence*. Setelah perulangan selesai akan didapatkan keluaran jumlah nilai 1 pada kelas *bully* dan bukan *bully*.



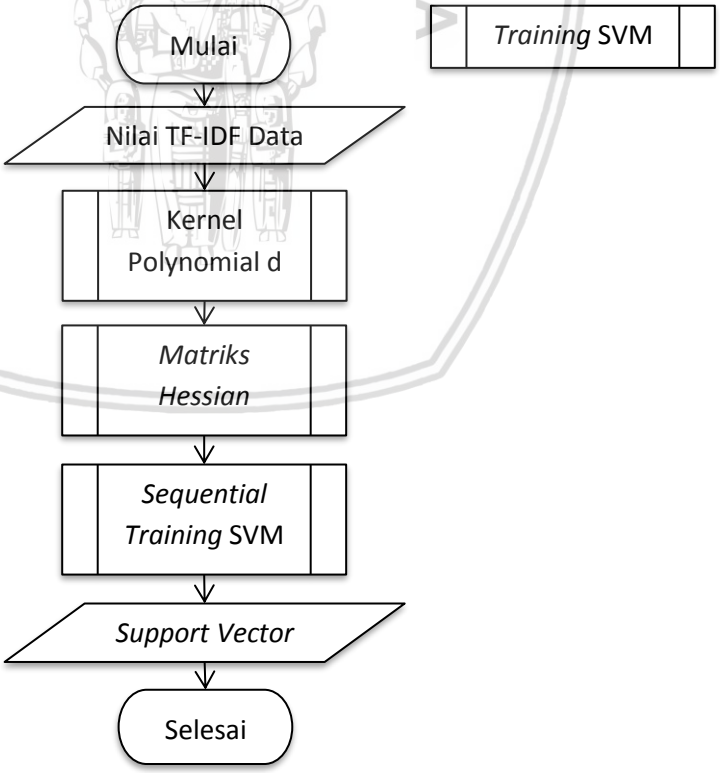
Gambar 4.16 Diagram Alir Hitung Nilai IG

Proses terakhir dalam seleksi fitur yaitu menghitunga nilai IG masing-masing *term*. Alur kerja perhitungan nilai IG ditunjukan pada diagram alir Gambar 4.16. Input dari proses ini adalah jumlah nilai 1 dan 0 pada dokumen dan jumlah nilai 1 pada tiap kelas. Kemudian masuk ke perulangan dan proses menghitunga nilai IG. Keluaran dari proses ini adalah nilai IG masing-masing *term*.



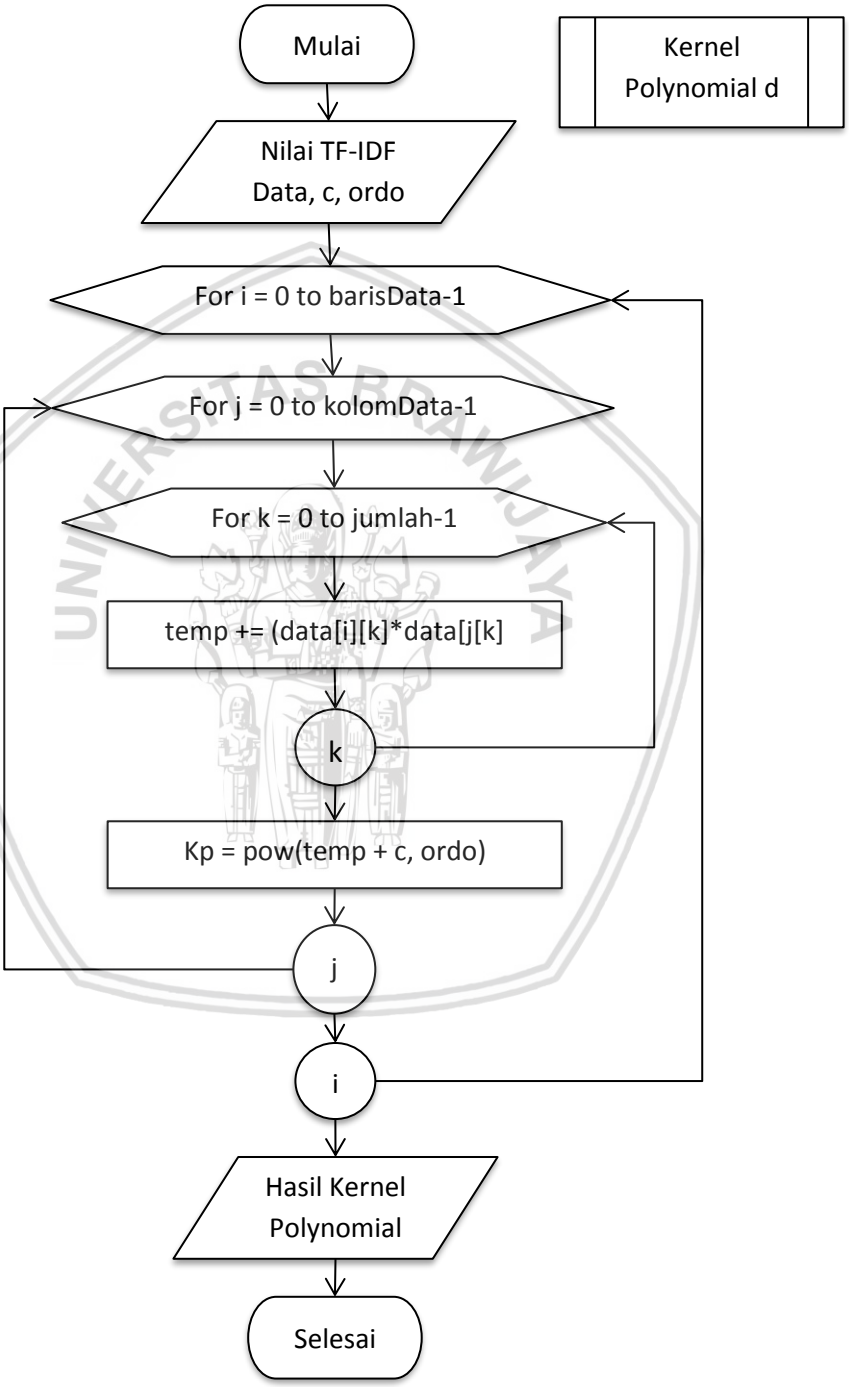
Gambar 4.17 Diagram Alir Proses SVM

Proses ketiga dalam sistem adalah identifikasi *cyberbullying* yang ditunjukan pada diagram alir Gambar 4.17. *Input* dari proses ini adalah nilai *term* TF-IDF yang telah di seleksi fitur. Proses identifikasi menggunakan metode SVM, yang terdiri dari 2 proses yaitu *traning* SVM dan *testing* SVM. Keluaran dari proses ini adalah identifikasi *tweet bullying*.



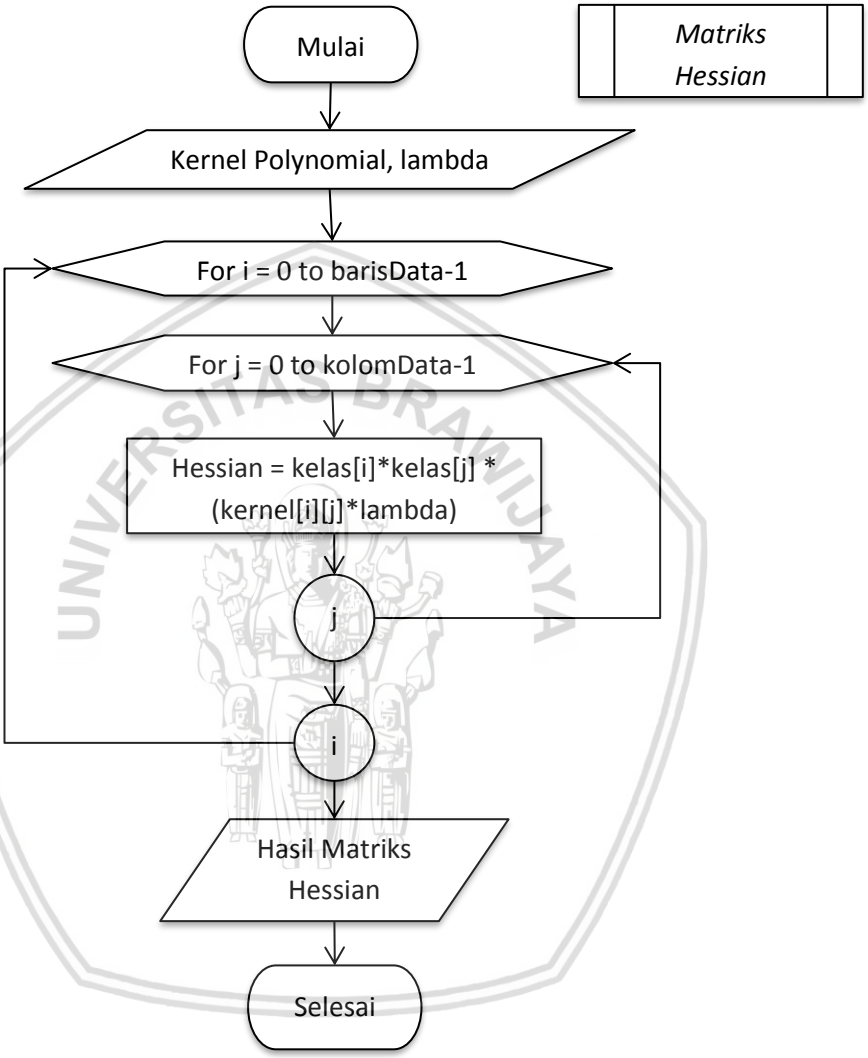
Gambar 4.18 Diagram Alir Training SVM

Proses *training* SVM ditunjukan pada diagram alir Gambar 4.18. *Input* dari *training* metode SVM merupakan nilai TF-IDF dari *term*. Proses klasifikasi dimulai dari menghitung nilai kernel polynomial d, kemudian menghitung nilai *Matriks Hessian*. Setelah itu melakukan *Sequential Training* SVM dengan menghitung nilai E_i, $\delta\alpha$, dan α_i. Keluaran dari proses *training* SVM adalah model *support vector* untuk proses *testing* SVM.



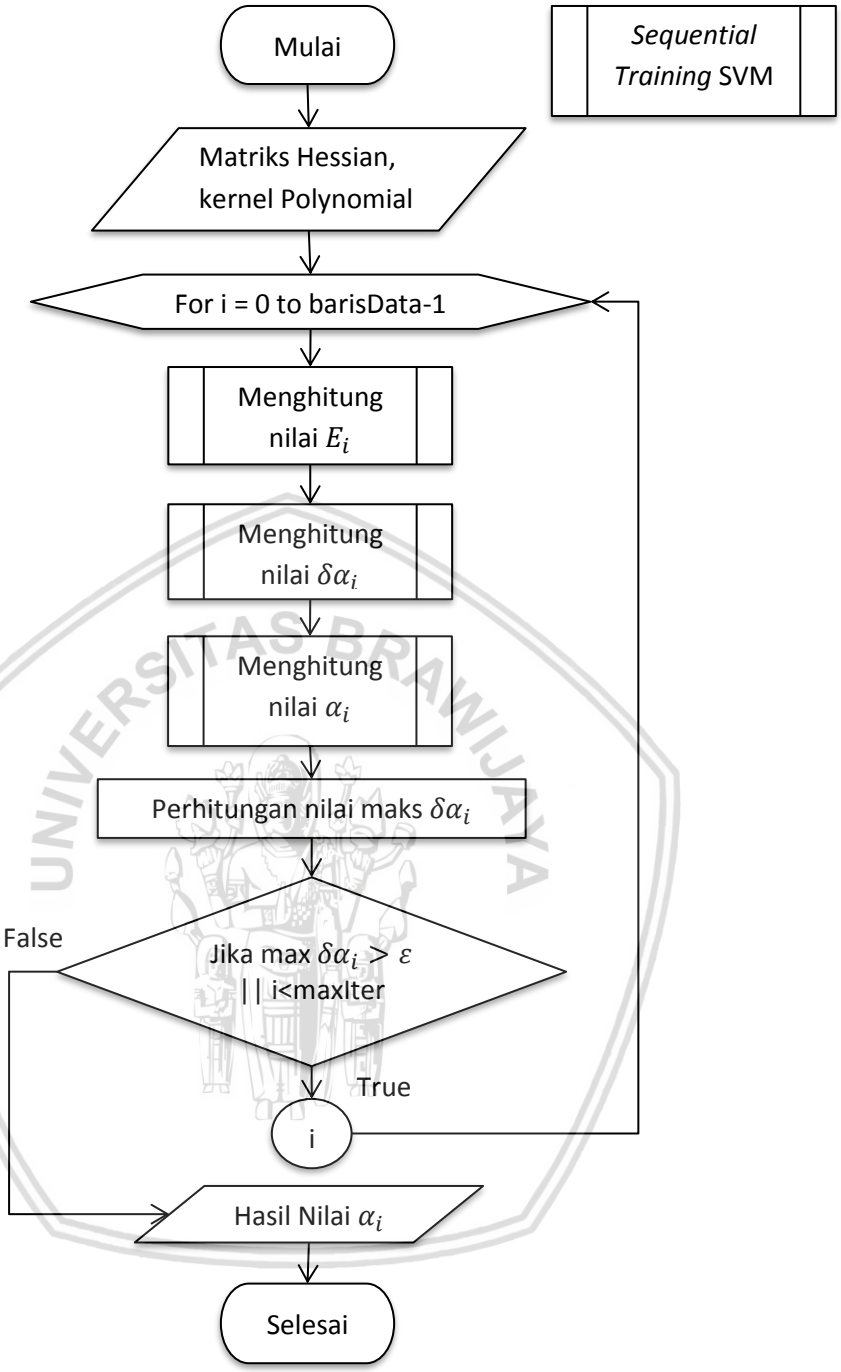
Gambar 4.19 Perhitungan Kernel Polynomial

Proses perhitungan Kernel Polynomial dimulai dari melakukan *input* data *training* dari hasil *text preprocessing* yang ditunjukan pada diagram alir Gambar 4.19. Kemudian melakukan perulangan hingga banyaknya data *training* dan masuk ke proses perhitungan kernel. Pada penelitian ini polynomial degree d dipangkatkan dengan ordo $d = 2$. Setelah itu didapatkan hasil berupa *Matriks* Kernel.



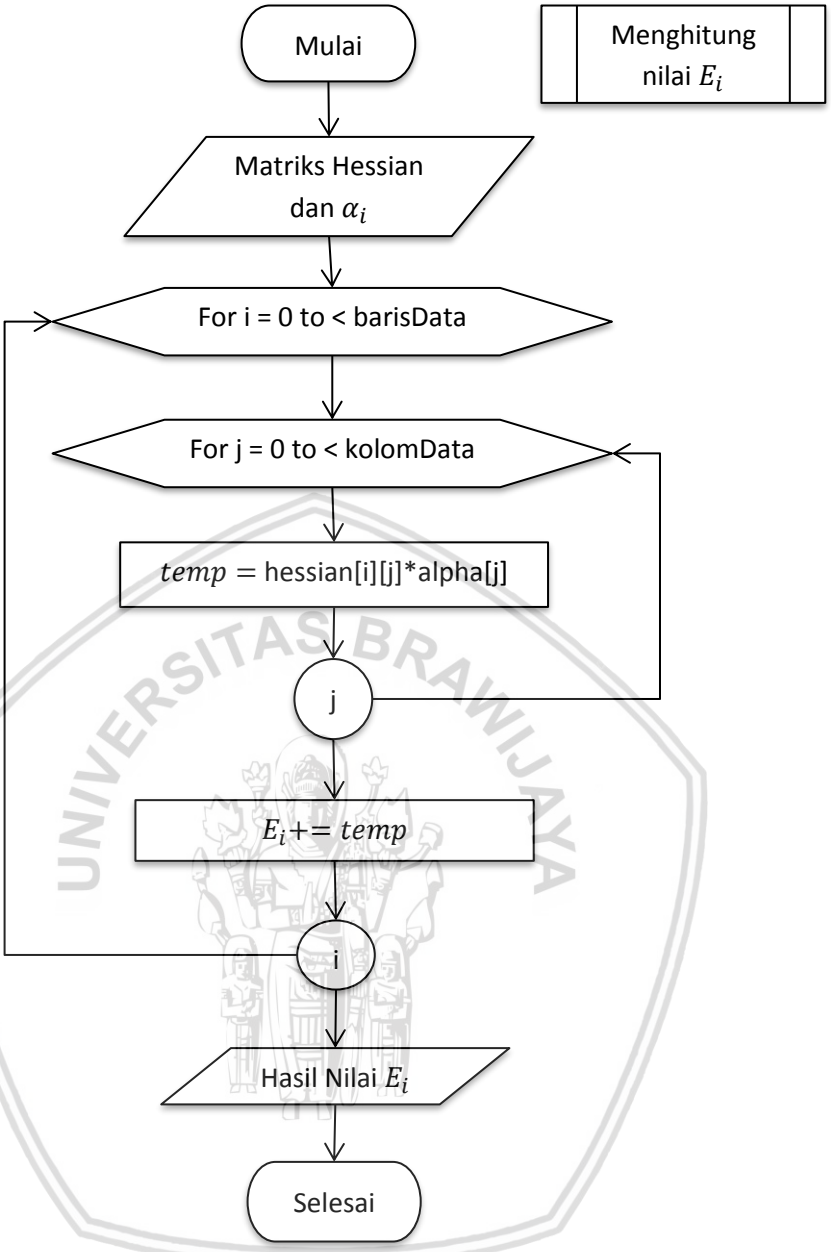
Gambar 4.20 Perhitungan Matriks Hessian

Proses perhitungan *Matriks Hessian* dilakukan setelah *Matriks* Kernel Polynomial terbentuk, alur kerja perhitungan ditunjukan pada diagram alir Gambar 4.20. *Input* dari proses perhitungan *Matriks Hessian* adalah nilai hasil kernel Polynomial. Kemudian melakukan perulangan dan proses menghitung *Matriks Hessian*. Setelah perulangan selesai didapatkan nilai *Matriks Hessian*.



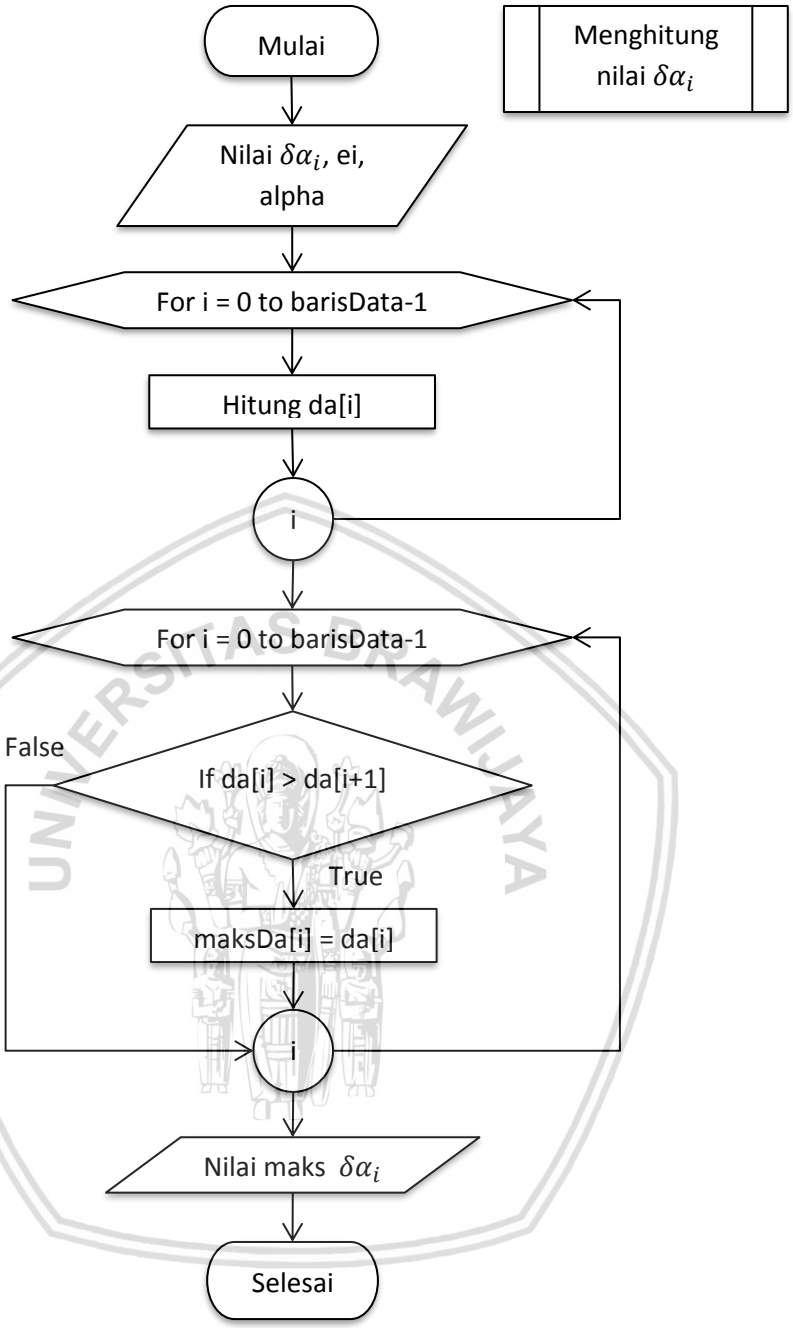
Gambar 4.21 *Sequential Training* SVM

Proses selanjutnya adalah perhitungan sequential training SVM, yang ditunjukan pada diagram alir Gambar 4.21. Perhitungan *Sequential Training* SVM dilakukan dalam tiga tahap perhitungan yang diulang sejumlah maksimum iterasi yang telah ditentukan. Tahapan dalam proses *Sequential Training* yaitu melakukan perhitungan mencari nilai E_i untuk tiap data *training*, menghitung nilai $\delta\alpha$, dan menghitung nilai α_i. *Input* dari proses ini adalah hasil dari perhitungan kernel dan *Matriks Hessian*. Dan keluaran dari proses ini adalah nilai α_i baru.



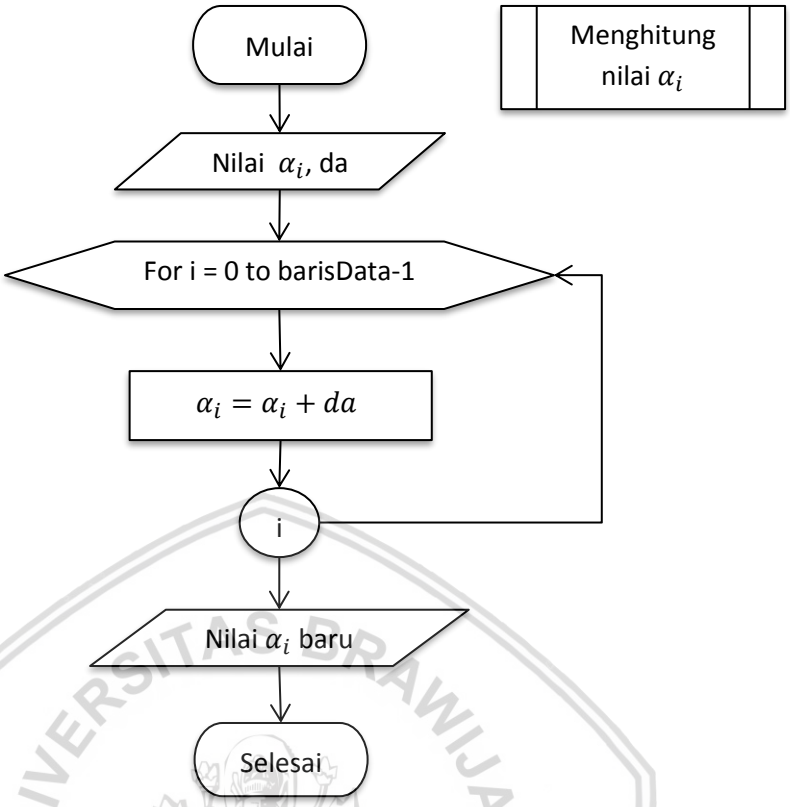
Gambar 4.22 Perhitungan Nilai E_i

Tahap pertama sequential training SVM yaitu menghitung nilai E_i, yang ditunjukan pada diagram alir Gambar 4.22. Perhitungan nilai E_i dilakukan dengan menjumlahkan hasil perkalian matriks hessian dengan nilai alpha ke-i.



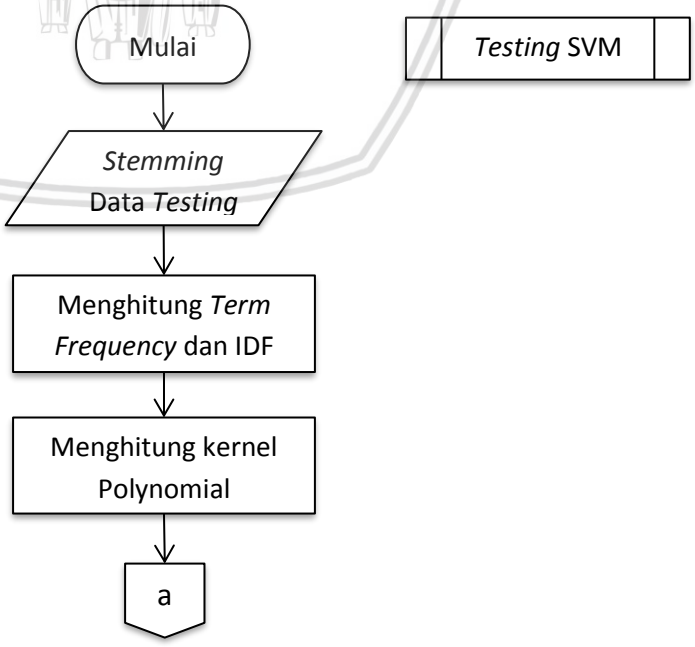
Gambar 4.23 Perhitungan Nilai Delta Alpha

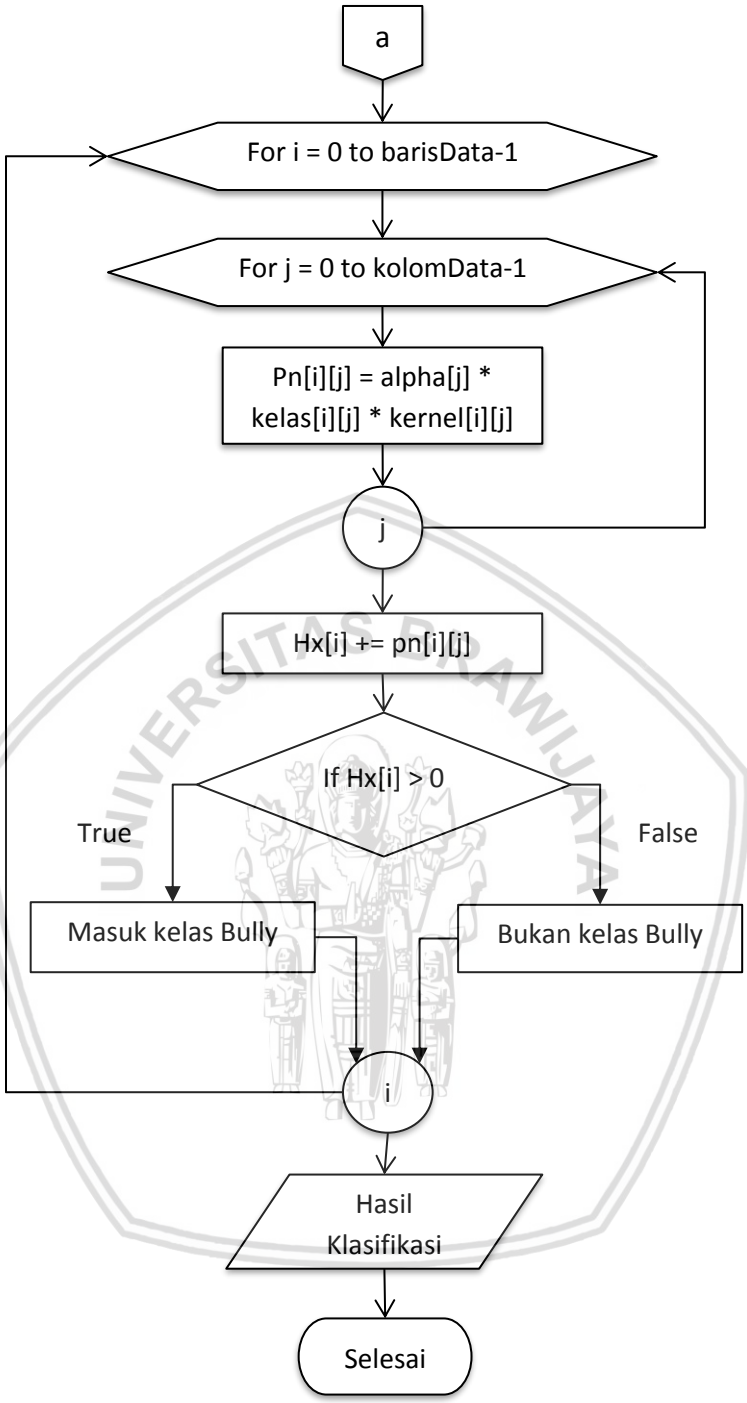
Tahap kedua yaitu perhitungan nilai $\delta\alpha$ (*delta alpha*) yang ditunjukan pada diagram alir Gambar 4.23. Perhitungan yang dilakukan untuk mendapatkan nilai *da[i]* (delta alpha) dengan mengkalikan *gamma* dengan hasil dari 1 dikurang nilai E_i ke-i dan dikurang dengan nilai *alpha* ke-i untuk mencari nilai maksimum. Hasil nilai maksimum akan dibandingkan dengan nilai C dikurang alpha ke-i. Kemudian mencari nilai maksimum *delta alpha*.



Gambar 4.24 Perhitungan Nilai Alpha ke-i

Tahap ketiga yaitu perhitungan nilai α_i (*alpha* ke-*i*), yang ditunjukan pada diagram alir Gambar 4.24. Perhitungan yang dilakukan dengan menambahkan nilai delta *alpha* ke-*i* dengan nilai *alpha* ke-*i* sebelumnya. Keluaran dari proses ini adalah nilai *alpha* ke-*i* yang baru.





Gambar 4.25 Proses Testing SVM

Proses kedua dalam perhitungan SVM adalah testing SVM, yang ditunjukan pada diagram alir Gambar 4.25. Tahap pertama yang dilakukan yaitu menghitung *term frequency* data *testing* berdasarkan *term* dari hasil *text preprocessing* data *training*, kemudian menghitung Kernel dari data *testing* dan data *training*. Setelah itu menghitung nilai Hx yang merupakan hasil penjumlahan nilai bobot kelas yang positif dan negatif, dan yang terakhir klasifikasi kelas jika positif maka dokumen merupakan *cyberbullying* dan jika tidak maka bukan *cyberbullying*.

4.5 Perancangan Pengujian

Pengujian yang dilakukan adalah pengujian parameter *Support Vector Machine* (SVM) dan pengujian *threshold* seleksi fitur *Information Gain* (IG). Parameter SVM yang diubah secara manual untuk mengetahui perubahan hasil identifikasi yang didapatkan. Untuk pengujian parameter seleksi fitur IG menggunakan sepuluh *threshold*. Dengan begitu dapat menganalisis pengaruh dari parameter seleksi fitur IG untuk identifikasi *tweet cyberbullying*.

4.5.1 Perancangan Pengujian Parameter Sequential Training SVM

Pengujian terhadap parameter *Sequential Training* SVM dilakukan untuk mengetahui pada skenario mana yang mengahasilkan nilai terbaik. Untuk skenario parameter nilai λ (*Lambda*) yaitu 0.1, 0.3, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, dan 4. Niai *default lambda* yang digunakan yaitu 0.5. Perancangan pengujian yang digunakan ditunjukan pada Tabel 4.31.

Tabel 4.31 Pengujian Pengaruh Nilai *Lambda*

Lambda	**Hasil**			
	Accuracy	***Precision***	***Re-call***	***F-Measure***
0.1				
0.3				
0.5				
1				
1.5				
2				
2.5				
3				
3.5				
4				

Pengujian selanjutnya adalah skenario untuk parameter nilai ϒ (konstanta *gamma*). Skenario pengujian yang dilakukan yaitu 0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 1.5, 2, dan 10. Nilai default konstanta *gamma* yang digunakan yaitu 0.001. Pengujian parameter konstanta *gamma* ditunjukan pada Tabel 4.32.

Tabel 4.32 Pengujian Pengaruh Nilai Konstanta *Gamma*

Konstanta *Gamma*	**Hasil**			
	Accuracy	***Precision***	***Re-call***	***F-Measure***
0.001				
0.01				
0.05				
0.1				

0.2				
0.5				
1				
1.5				
2				
10				

Pengujian selanjutnya adalah skenario untuk parameter nilai ε (*epsilon*). Skenario pengujian yang dilakukan yaitu 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, dan 100. Nilai default *epsilon* yang digunakan yaitu 0.0001. Pengujian parameter *epsilon* ditunjukan pada Tabel 3.33.

Tabel 4.33 Pengujian Pengaruh Nilai *Epsilon*

Epsilon	**Hasil**			
	Accuracy	***Precision***	***Re-call***	***F-Measure***
0.0000001				
0.000001				
0.00001				
0.0001				
0.001				
0.01				
0.1				
1				
10				
100				

Pengujian selanjutnya adalah skenario untuk parameter nilai C (*Complexity*). Skenario pengujian yang dilakukan yaitu 1, 10, 20, 30, 40, 50, 60, 70, 80 dan 90. Nilai default C yang digunakan yaitu 1. Pengujian parameter C ditunjukan pada Tabel 3.34.

Tabel 4.34 Pengujian Pengaruh Nilai C

C	**Hasil**			
	Accuracy	***Precision***	***Re-call***	***F-Measure***
1				
10				
20				
30				
40				

50				
60				
70				
80				
90				

Pengujian selanjutnya adalah skenario untuk nilai maksimum iterasi. Skenario pengujian yang dilakukan yaitu dengan maksimum iterasi (*iterMax*) 3, 10, 100, 200, 500, 1000, 5000, 10000, 50000, dan 100000. Nilai default maksimum iterasi yang digunakan yaitu sejumlah 3. Pengujian maksimum iterasi ditunjukan pada Tabel 4.35.

Tabel 4.35 Pengujian Maksimum Iterasi

Iterasi	**Hasil**			
	Accuracy	***Precision***	***Re-call***	***F-Measure***
10				
20				
50				
100				
200				
500				
1000				
2000				
5000				
10000				

4.5.2 Perancangan Pengujian Threshold Information Gain

Perancangan pengujian untuk parameter *Information Gain* terbagi atas *threshold* yaitu 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% dan 100%. Hal ini dilakukan untuk mengetahui pengaruh dari parameter *threshold* dari seleksi fitur *Information Gain*. Skenario pengujian *threshold* IG ditunjukan pada Tabel 4.36.

Tabel 4.36 Pengujian *Threshold Information Gain*

Threshold	**Hasil**			
	Accuracy	***Precision***	***Re-call***	***F-Measure***
10%				
20%				
30%				
40%				

50%				
60%				
70%				
80%				
90%				
100%				

BAB 5 HASIL

Pada bab ini menjelaskan implementasi algoritme dari perancangan yang telah dilakukan pada bab sebelumnya. Implementasi yang ditampilkan berupa potongan kode program beserta penjelasan program tiap baris.

5.1 Implementasi Algoritme

Implementasi algoritme metode SVM dan seleksi fitur IG untuk identifikasi *tweet cyberbullying* menerapkan algoritme seperti yang dijelaskan pada diagram alir sistem.

5.1.1 Implementasi *Text Preprocessing*

Text preprocessing merupakan tahap awal dalam implementasi sistem, diamana dalam tahap ini akan mengubah data yang tidak terstruktur menjadi terstruktur. *Text preprocessing* dibagi menjadi beberapa tahapan yaitu tokenisasi, *filtering*, *stemming* dan *term weighting*.

5.1.1.1 Implementasi Algoritme Tokenisasi

Tokenisasi adalah proses untuk memisahkan kata dan membuang tanda baca dan angka serta mengubah kata menjadi huruf kecil. Potongan kode program untuk proses tokenisasi ditunjukan pada Kode Program 5.1.

Preprocessing.java

```
1   public String caseFolding(String inputFile) throws
    FileNotFoundException, IOException {
2       return inputFile.toLowerCase();
3   }
4
5   public String hapusAngkadanTandaBaca(String kalimat) {
6       kalimat = kalimat.replaceAll("\\p{Punct}", "");
7       return kalimat;
8   }
9
10  public ArrayList<String> token(String kalimat) throws
    IOException {
11      String hasilKalimat = hapusAngkadanTandaBaca(kalimat);
12      String hasilKalimat2 = caseFolding(hasilKalimat);
13      String hapusNewline =
    hasilKalimat2.replace(System.getProperty("line.separator"),
    " ");
14      listKata = new ArrayList<String>();
15      token = new StringTokenizer(hapusNewline, " ");
16      while (token.hasMoreTokens()) {
17          words = token.nextToken();
18          listKata.add(words);
19      }
20      return listKata;
21  }
```

Kode Program 5.1 Tokenisasi

Keterangan Kode Program 5.1:

Baris 1-3 : deklarasi *method caseFolding*() dengan *input String* yang mengembalikan nilai *input* menjadi huruf kecil.

Baris 5-8 : deklarasi *method hapusAngkadanTandaBaca*() dengan *input String* yang berisi proses menghapus nilai *input* yang ada tanda baca, dan mengembalikan nilai *input*.

Baris 10-21 : deklarasi *method ArrayList token*() dengan *input String*.

Baris 11 : inisialisasi variabel *String* hasilKalimat dengan nilai memanggil *method hapusAngkadanTandaBaca*() dan parameter nilai *input*.

Baris 12 : inisialisasi variabel *String* hasilKalimat2 dengan nilai memanggil *method caseFolding*() dan parameter nilai variabel hasilKalimat.

Baris 13 : inisialisasi variabel *String* hapusNewLine dengan nilai variabel hasilKalimat2 yang dipisahkan oleh spasi.

Baris 16-19 : melakukan perulangan dengan kondisi jika variabel token masih memiliki token maka nilai token berikutnya dimasukan ke variabel *words* kemudian ditambahkan ke *ArrayList* listKata.

Baris 20 : mengembalikan variabel listKata.

5.1.1.2 Implementasi Algoritme *Filtering*

Filtering merupakan proses membuang kata yang tidak penting dengan menggunakan kamus *stopword*. Potongan kode program untuk proses *filtering* ditunjukan pada Kode Program 5.2.

Preprocessing.java

```
1  public String bersihkanStopword(String inputdokumen)
   throws FileNotFoundException, IOException {
2      String daftarstopword = readDocument("Stopwords.txt");
3      String stop = "\\b(" + daftarstopword + ")\\b\\s*";
4      Pattern stopword = Pattern.compile(stop,
   Pattern.CASE_INSENSITIVE);
5      Matcher sesuai = stopword.matcher(inputdokumen);
6      bersih = sesuai.replaceAll("");
7      return bersih;
8  }
```

Kode Program 5.2 *Filtering*

Keterangan Kode Program 5.2:

Baris 1-8 : deklarasi *method bersihkanStopword*() dengan *input String*.

Baris 2 : inisialisasi variabel *String* daftarstopword dengan memanggil *method redDocument*() untuk membaca file *Stopwords* bertipe txt.

Baris 3 : inisialisasi variabel *String* stop dengan list *stopword*.

Baris 4 : inisialisasi variabel *Pattern* stopword dengan memanggil *method compile*().

Baris 5 : inisialisasi variabel *Matcher* sesuai dengan nilai variabel *stopword* yang sesuai dengan list pada *input*.

Baris 6 : mengahpus nilai yang sama pada variabel sesuai ke dalam variabel bersih.

Baris 7 : mengembalikan nilai variabel bersih.

5.1.1.3 Implementasi Algoritme *Stemming*

Stemming merupakan proses mencari kata dasar dari sebuah kata, dengan menggunakan *stemmer library* Nazief & Adriani. Potongan kode program untuk proses *stemming* ditunjukan pada Kode Program 5.3.

Preprocessing.java

```
1   public ArrayList<String> stemming(ArrayList<String>
    kalimat) throws IOException {
2       StemmingNazief & Adriani stem = new StemmingNazief &
    Adriani();
3       for (int i = 0; i < kalimat.size(); i++) {
4           kalimat.set(i, stem.KataDasar(kalimat.get(i)));
5       }
6       return kalimat;
7   }
8
9   public String kataDasar(String kalimat) throws IOException
    {
10      String out = "";
11      stemming(token(kalimat));
12      for (int i = 0; i < listKata.size(); i++) {
13          out += listKata.get(i) + " ";
14      }
15      return out;
16  }
```

Kode Program 5.3 *Stemming*

Keterangan Kode Program 5.3:

Baris 1-7 : deklarasi *method stemming*() dengan *input ArrayList String*.

Baris 2 : instansiasi *object* stem dari kelas StemmingNazief & Adriani.

Baris 3-5 : melakukan perulangan dengan mengeset nilai *input* dan memanggil *method kataDasar*().

Baris 6 : mengembalikan nilai kalimat.

Baris 9-16 : deklarasi *method kataDasar*() dengan *input String*.

Baris 10 : inisialisasi variabel *String* out.

Baris 11 : memanggil *method stemming*() dengan parameter memanggil *method* token dengan parameter nilai *input*.

Baris 12-14 : melakukan perulangan dengan variabel out bernilai *ArrayList* listKata ke-i.

Baris 15 : mengembalikan nilai out.

5.1.1.4 Implementasi Algoritme *Term Weighting*

Proses *term weighting* terbagi atas beberapa tahap yaitu *term frequency, weight term frequency, document frequency, inverse document frequency* dan *weight term document*. Potongan kode program untuk proses term weighting ditunjukan pada Kode Program 5.4.

```
Preprocessing.java
```

```
1   public HashMap frequency(ArrayList<String> input) throws
    FileNotFoundException {
2       map = new HashMap<String, Integer>();
3       tokenClean = new ArrayList<String>();
4       for (int i = 0; i < input.size(); i++) {
5           String kata = input.get(i);
6           Integer jumlah = map.get(kata);
7           if (map.containsKey(kata)) {
8               map.put(kata, jumlah + 1);
9               if (jumlah == 1) {
10                  tokenClean.add(kata);
11              }
12          } else {
13              map.put(kata, 1);
14              tokenClean.add(kata);
15          }
16      }
17      return map;
18  }
19
20  public double[][] hitungTF(ArrayList<String> kalimat,
    List<ArrayList<String>> data, int dokumen) throws
    FileNotFoundException {
21      double[][] tf = new double[kalimat.size()][dokumen];
22      for (int i = 0; i < kalimat.size(); i++) {
23          for (int j = 0; j < dokumen; j++) {
24              ArrayList<String> word = data.get(j);
25              for (int k = 0; k < word.size(); k++) {
26                  String kata = word.get(k);
27                  if (kalimat.get(i).contains(kata)) {
28                      tf[i][j] += 1;
29                  }
30              }
31          }
32      }
33      return tf;
34  }
35
36  public double[][] hitungWTF(ArrayList<String> kalimat,
    double[][] tf, int dokumen) {
37          double[][] wtf = new
    double[kalimat.size()][dokumen];
38      for (int i = 0; i < kalimat.size(); i++) {
39          for (int j = 0; j < dokumen; j++) {
40              if (tf[i][j] == 0) {
41                  wtf[i][j] = 0;
42              } else {
43                  wtf[i][j] = 1 + Math.log10(tf[i][j]);
44              }
45          }
46      }
47      return wtf;
48  }
49
50  public double[] hitungDF(ArrayList<String> kalimat,
    double[][] tf, int dokumen) {
51      double[] df = new double[kalimat.size()];
52      for (int i = 0; i < kalimat.size(); i++) {
53          for (int j = 0; j < dokumen; j++) {
```

```
54              if (tf[i][j]>0) {
55                  df[i] += 1;
56              }
57          }
58      }
59      return df;
60  }
61
62  public double[] hitungIDF(double[] df,
    List<ArrayList<String>> data) {
63      double[] idf = new double[df.length];
64      for (int i = 0; i < df.length; i++) {
65          if (df[i] == 0) {
66              idf[i] = 0;
67          } else {
68              idf[i] = Math.log10(data.size() / df[i]);
69          }
70      }
71      return idf;
72  }
73
74  public double[][] hitungWTD(ArrayList<String> kalimat,
    double[][] tf, double[] idf, int dokumen) {
75      double[][] wtd = new double[kalimat.size()][dokumen];
76      for (int i = 0; i < kalimat.size(); i++) {
78          for (int j = 0; j < dokumen; j++) {
79              wtd[i][j] = tf[i][j] * idf[i];
80          }
81      }
82      return wtd;
83  }
```

Kode Program 5.4 *Term Weighting*

Keterangan Kode Program 5.4:

Baris 1-18 : deklarasi *method frequency*() dengan *input ArrayList String*. *Method* berguna untuk mengetahui *frequency* semua kata.

Baris 2 : inisialisasi variabel map bertipe HashMap.

Baris 3 : inisialisasi variabel tokenClean yang berguna menampung token untuk proses selanjutnya.

Baris 4-16 : melakukan perulangan untuk menghitung jumlah frequency variabel *input*.

Baris 17 : mengembalikan nilai variabel map.

Baris 20-34 : deklarasi *method hitungTF*() dengan *input ArrayList String* kalimat, data dan jumlah dokumen. *Method* berguna untuk menhitung *term frequency* pada setiap dokumen.

Baris 21 : inisialisasi variabel tf *array* dua dimensi dengan jumlah baris baris sebanyak banyak *input* dan jumlah kolom sebanyak variabel dok.

Baris 22-32 : melakukan perulangan untuk menghitung *frequency* kemunculan kata.

Baris 24 : inisialisasi variabel *ArrayList String word* dengan nilai memanggil method tokenDokumen().

Baris 25-30 : melakukan perulangan untuk menghitung jumlah kata yang ada dalam variabel word.

Baris 33 : mengembalikan nilai tf.

Baris 36-48 : deklarasi *method hitungWTF*() dengan *input* kalimat, nilai tf dan jumlah dokumen. *Method* berguna untuk menghitung weight term frequency.

Baris 37 : inisialisasi variabel wtf dengan jumlah baris sebanyak jumlah kalimat dan jumlah kolom sebanyak jumlah dok.

Baris 38-46 : melakukan perulangan untuk menghitung nilai wtf, dengan kondisi jika tf bernilai 0 maka hasil wtf sama dengan 0, dan jika selain 0 maka dihitung sesuai rumus wtf.

Baris 47 : mengembalikan nilai wtf.

Baris 50-60 : deklarasi *method hitungDF*() dengan parameter kalimat, hasil tf dan jumlah dokumen. *Method* berguna untuk menghitung *document frequency*.

Baris 51 : inisialisasi variabel df dengan jumlah baris sebanyak jumlah kalimat.

Baris 52-58 : melakukan perulangan dengan kondisi jika nilai tf lebih dari 0 maka term muncul didokumen tersebut sehingga mendapatkan mendapatkan nilai df.

Baris 59 : mengembalikan nilai df.

Baris 62-72 : deklarasi *method hitungIDF*() dengan parameter hasil df dan data. *Method* berguna untuk menghitung *inverse document frequency*.

Baris 63 : inisialisasi variabel idf dengan jumlah baris sebanyak jumlah df.

Baris 64-70 : melakukan perulangan untuk menghitung idf dengan kondisi jika nilai df sama dengan 0 maka idf bernilai 0, sedangkan jika selain 0 maka dihitung menggunakan rumus idf.

Baris 71 : mengembalikan nilai idf.

Baris 74-83 : deklarasi *method hitungWTD*() dengan parameter kalimat, hasil tf, hasil idf dan jumlah dokumen. *Method* berguna untuk menghitung nilai weight term frequency.

Baris 75 : inisialisasi variabel wtd dengan jumlah baris sebanyak jumlah kalimat dan jumlah kolom sebanyak jumlah dokumen.

Baris 76-81 : melakukan perulangan untuk menghitung nilai wtd.

Baris 82 : mengembalikan nilai wtd.

5.1.2 Implementasi *Information Gain*

Tahap kedua dalam sistem ini yaitu seleksi fitur mengunakan *Information Gain* yang terbagi atas beberapa tahap yaitu hitung *term presence* dan kemudian menghitung nilai *entropy*.

5.1.2.1 Implementasi Algoritme *Term Presence*

Proses perhitungan *term presence* yaitu menghitung apakah suatu *term* muncul dalam suatu dokumen. Potongan kode program untuk proses *term presence* ditunjukan pada Kode Program 5.5

InformationGain.java

```
1   public double[][] hitungTP(ArrayList<String> kalimat,
    List<ArrayList<String>> data, int dokumen) {
2       double[][] tp = new double[kalimat.size()][dokumen];
3       for (int i = 0; i < kalimat.size(); i++) {
4           for (int j = 0; j < dokumen; j++) {
5               ArrayList<String> word =
    tokenDokumen(data.get(j).get(0).toString());
6               for (int k = 0; k < word.size(); k++) {
7                   String kata = word.get(k);
8                   if (kata.contains(kalimat.get(i))) {
9                       tp[i][j] = 1;
10                  }
11              }
12          }
13      }
14      return tp;
15  }
```

Kode Program 5.5 *Term Presence*

Keterangan Kode Program 5.5:

Baris 1 : deklarasi method hitungTP() dengan input ArrayList String kalimat, data dan jumlah dokumen. Method berguna untuk menghitung term presence.

Baris 2 : inisialisasi variabel tp dengan jumlah baris sebanyak jumlah kalimat dan jumlah kolom sebanyak jumlah dokumen.

Baris 3-13 : melakukan perulangan untuk menghitung term presence.

Baris 5 : inisialisasi variabel word dengan nilai memanggil method tokenDokumen().

Baris 6-11 : melakukan perulangan untuk menghitung jumlah term presence dengan kondisi jika kata terdapat pada kalimat maka tp bernilai 1.

Baris 14 : mengembalikan nilai tp.

5.1.2.2 Implementasi Algoritme Menghitung *Information* Gain

Proses perhitungan nilai *information gain* terbagi atas beberapa tahap, dimulai dari menghitung kemunculan nilai 1, menghitung nilai 1 dan 0 pada tiap

kelas, serta menghitung nilai *entropy* tiap fitur dokumen. Potongan kode program untuk proses perhitungan *information gain* ditunjukan pada Kode Program 5.6.

InformationGain.java

```
1   public double[][] hitungJumlah01(ArrayList<String>
    kalimat, double[][] tp, int dokumen) {
2       double[][] jumlah01 = new double[kalimat.size()][2];
3       for (int i = 0; i < kalimat.size(); i++) {
4           for (int j = 0; j < dokumen; j++) {
5               if (tp[i][j] == 0) {
6                   jumlah01[i][0] += 1;
7               } else {
8                   jumlah01[i][1] += 1;
9               }
10          }
11      }
12      return jumlah01;
13  }
14
15  public double[][] hitungJumlah1Kelas(ArrayList<String>
    kalimat, List<ArrayList<String>> data, double[][] tp, int
    dokumen) {
16      double[][] jumlah1Kelas = new
    double[kalimat.size()][2];
17      for (int i = 0; i < kalimat.size(); i++) {
18          for (int j = 0; j < dokumen; j++) {
19              if (data.get(j).get(1).equals("No")) {
20                  jumlah1Kelas[i][0] += tp[i][j];
21              } else if (data.get(j).get(1).equals("Yes"))
22  {
23                  jumlah1Kelas[i][1] += tp[i][j];
24              }
25          }
26      }
27      return jumlah1Kelas;
28  }
29
    public double[] hitungIG(ArrayList<String> kalimat,
    List<ArrayList<String>> data, double[][] jumlah01,
30  double[][] jumlah1Kelas, int dokumen) {
31      double[] ig = new double[kalimat.size()];
        double yes = 0, no = 0, jumlah = data.size(), nilai1
32  = 0;
33      for (int i = 0; i < dokumen; i++) {
34          if (data.get(i).get(1).equals("No")) {
35              no += 1;
36          } else if (data.get(i).get(1).equals("Yes")) {
37              yes += 1;
38          }
39      }
40      for (int i = 0; i < kalimat.size(); i++) {
41          nilai1 += jumlah01[i][1];
42      }
43      double yes1, yes0, no1, no0, y = 0, n =0;
44      for (int i = 0; i < kalimat.size(); i++) {
45          if (jumlah1Kelas[i][1] != 0) {
                yes1 = ((jumlah1Kelas[i][1] / yes) *
46  Math.log10(jumlah1Kelas[i][1] / yes));
47          } else {
48              yes1 = 0;
49          }
50          if (jumlah1Kelas[i][0] != 0) {
```

```
51              yes0 = ((jumlah1Kelas[i][0]) / no *
52  Math.log10((jumlah1Kelas[i][0]) / no));
53          } else {
54              yes0 = 0;
            }
55          if (yes-jumlah1Kelas[i][1] != 0) {
56              no1 = (((yes-jumlah1Kelas[i][1]) / yes) *
57  Math.log10((yes-jumlah1Kelas[i][1]) / yes));
58          } else {
59              no1 = 0;
60          }
            if (no - jumlah1Kelas[i][0] != 0) {
61              no0 = ((no - jumlah1Kelas[i][0]) / no *
62  Math.log10((no - jumlah1Kelas[i][0]) / no));
63          } else {
64              no0 = 0;
65          }
66          if (jumlah01[i][1] != 0) {
67              y = jumlah01[i][1];
68          }
69          if (jumlah01[i][0] != 0) {
70              n = jumlah01[i][0];
            }
            ig[i] = -(yes / jumlah * Math.log10(yes / jumlah)
    + (no / jumlah * Math.log10(no / jumlah)))
71                  + ((y / nilai1) * (yes1 + yes0))
72                  + ((n / nilai1) * (no1 + no0));
73      }
74      return ig;
75  }

76  public Object[][] sorting(ArrayList<String> kalimat,
77  double[] ig) {
78      double[] sortIG = new double[kalimat.size()];
79      String[] sortFitur = new String[kalimat.size()];
80      Object[][] sorting = new Object[kalimat.size()][2];
81      sortIG = ig;
82      for (int i = 0; i < kalimat.size(); i++) {
83          sortFitur[i] = kalimat.get(i);
84      }
85      for (int i = 0; i < kalimat.size(); i++) {
86          for (int j = 0; j < kalimat.size() - 1; j++) {
87              if (sortIG[j] < sortIG[j + 1]) {
88                  double temp = sortIG[j];
89                  sortIG[j] = sortIG[j + 1];
90                  sortIG[j + 1] = temp;
91                  String tempK = sortFitur[j];
92                  sortFitur[j] = sortFitur[j + 1];
93                  sortFitur[j + 1] = tempK;
94              }
95          }
96      }
97      for (int i = 0; i < kalimat.size(); i++) {
98          sorting[i][0] = sortFitur[i];
99          sorting[i][1] = sortIG[i];
100     }
101     return sorting;
102 }
```

```
103 public Object[][] hitungSeleksiFitur(ArrayList<String>
    kalimat, Object[][] sorting, double threshold) {
104     Object[][] seleksiFitur = new
105 Object[kalimat.size()][2];
106     for (int i = 0; i < kalimat.size(); i++) {
107         seleksiFitur[i][0] = sorting[i][0];
108         seleksiFitur[i][1] = sorting[i][1];
    }
109     int jumlah = (int) Math.round(kalimat.size() *
110 threshold);
    for (int i = 0; i < jumlah; i++) {
111         System.out.println(seleksiFitur[i][0] + " - " +
112 seleksiFitur[i][1]);
113     }
    return seleksiFitur;
}
```

Kode Program 5.6 *Information Gain*

Keterangan Kode Program 5.6:

Baris 1-13 : deklarasi *method hitungJumlah01*() dengan *input ArrayList String* kalimat, hasil tp dan jumlah dokumen. *Method* berguna untuk menghitung jumlah nilai 0 dan 1 yang muncul pada *term presence*.

Baris 2 : inisialisasi variabel jumlah01 dengan jumlah baris sebanyak jumlah kalimat dan kolom sebanyak 2 untuk menampung nilai 0 dan 1.

Baris 3-11 : melakukan perulangan dengan kondisi jika tp bernilai 0 maka ditambahkan ke jumlah01 ke kolom 0 dan jika lainnya ditambahkan ke kolom 1.

Baris 12 : mengembalikan nilai jumlah01.

Baris 15-27 : deklarasi *method hitungJumlah1Kelas*() dengan parameter kalimat, data, hasil tp dan jumlah dokumen. *Method* berguna untuk menghitung jumlah nilai 1 pada tp pada setiap kelas.

Baris 16 : inisialisasi variabel jumlah1Kelas dengan jumlah baris sebanyak jumlah kalimat dan kolom sebanyak 2 untuk menampung kelas *bully* dan bukan.

Baris 17-25 : melakukan perulangan untuk menghitung jumlah nilai 1 dengan kondisi jika data masuk ke kelas *no* maka masuk ke kolom 0 dan jika *yes* maka masuk ke kolom 1.

Baris 26 : mengembalikan nilai jumlah1Kelas.

Baris 29-67 : deklarasi *method hitungIG*() dengan parameter kalimat, data, hasil jumlah01, hasil jumlah1Kelas dan jumlah dokumen. *Method* berguna untuk menghitung nilai information gain.

Baris 30 : inisialisasi variable ig dengan jumlah baris sebanyak jumlah kalimat.

Baris 31 : inisialisasi variable *yes*, *no*, jumlah dan nilai1.

Baris 32-38 : melakukan perulangan untuk menghitung jumlah kelas *no* dan kelas *yes* yang ditampung pada variabel *no* dan *yes*.

Baris 39-41 : melakukan perulangan untuk menghitung jumlah nilai 1 pada variabel jumlah01.

Baris 42 : inisialisasi variabel yes1, yes0, no1, no0. Variabel berguna untuk menampung jika nilai jumlah1kelas bukan 0.

Baris 43-71 : melakukan perulangan untuk menghitung nilai IG dengan beberapa kondisi agar ketika dihitung variabel yang bernilai 0 tidak menghasilkan nilai *infinity*.

Baris 72 : mengembalikan nilai ig.

Baris 75-100 : deklarasi *method sorting*() dengan parameter kalimat dan hasil ig. *Method* berguna untuk mengurutkan nilai ig dari terbesar ke kecil.

Baris 76 : inisialisasi variabel sortIG dengan jumlah baris sebanyak jumlah kalimat. Variabel berguna untuk menyimpan hasil pengurutan ig.

Baris 77 : inisialisasi variabel sortFitur dengan jumlah baris sebanyak jumlah kalimat. Variabel berguna untuk menyimpan kata yang diurutkan.

Baris 78 : inisialisasi variable *sorting* dengan jumlah baris sebahyak jumlah kalimat dan jumlah kolom sebanyak 2 untuk menampung hasil pengurutan fitur dan nilai ig.

Baris 80-82 : melakukan perulangan untuk menyimpan kalimat ke dalam variabel sortFitur.

Baris 83-94 : melakukan perulangan untuk mengurutkan variabel sortIG dan sortFitur.

Baris 95-98 : melakukan perulangan untuk menyimpan nilai hasil pengurutan sortIG dan sortFitur ke variable *sorting*.

Baris 99 : mengembalikan nila *sorting*.

Baris 102-113 : deklarasi *method seleksiFitur*() dengan parameter kalimat, hasil *sorting* dan nilai *threshold*. *Method* berguna untuk menghitung jumlah fitur sesaui *threshold* yang ditentukan.

Baris 103 : inisialisasi variabel seleksiFitur dengan jumlah baris sebanyak kalimat dan jumlah kolom sebanyak 2 untuk menampung kalimat dan nilai kalimat.

Baris 104-107 : melakukan perulangan untuk menyimpan nilai sortIG dan sortFitur ke dalam variabel seleksiFitur.

Baris 108 : inisialisasi variabel jumlah dengan nilai hasil perhitungan *threshold*. Variabel berguna untuk menampung jumlah *threshold* untuk seleksi fitur yang digunakan.

Baris 109-111 : melakukan perulangan untuk menampilkan hasil seleksi fitur.

Baris 112 : mengembalikan nilai seleksiFitur.

5.1.3 Implementasi *Support Vector Machine*

Tahap ketiga dari sistem ini adalah klasifikasi menggunakan metode SVM, dimana dalam perhitungan SVM terbagi atas beberapa tahap yaitu menghitungan kernel polynomial, matriks *Hessian* dan *sequential training* SVM.

5.1.3.1 Implementasi Algoritme Kernel Polynomial

Perhitungan kernel polynomial dilakukan dengan menghitung nilai dari hasil idf tiap *term* pada dokumen. Potongan kode program untuk proses perhitungan kernel polynomial ditunjukan pada Kode Program 5.7.

SVM.java

```
1   public double[][] kernelPolynomial(double[][] data1,
    double[][] data2, int baris, int kolom, int jumlah) {
2       double[][] kp = new double[baris][kolom];
3       for (int i = 0; i < baris; i++) {
4           for (int j = 0; j < kolom; j++) {
5               double temp = 0;
6               for (int k = 0; k < jumlah; k++) {
7                   temp += (data1[k][i] * data2[k][j]);
8               }
9               kp[i][j] = Math.pow(temp + c, ordo);
10          }
11      }
12      return kp;
13  }
```

Kode Program 5.7 Kernel Polynomial

Keterangan Kode Program 5.7:

Baris 1 : deklarasi *method kernelPolynomial*() dengan parameter data, jumlah baris dan kolom, dan jumlah hasil seleksi fitur. *Method* berguna untuk menghitung nilai matriks *kernel polynomial*.

Baris 2 : inisialisasi variabel kp dengan jumlah baris dan kolom sebanyak dokumen.

Baris 3-11 : melakukan perulangan untuk menghitung nilai *polynomial*.

Baris 12 : mengembalikan nilai kp.

5.1.3.2 Implementasi Algoritme Matriks Hessian

Proses perhitungan matriks *Hessian* dilakukan setelah hasil perhitungan *kernel polynomial* didapatkan. Potongan kode program untuk proses perhitungan matrik *Hessian* ditunjukan pada Kode Program 5.8.

SVM.java

```
1   public double[][] matriksHessian(double[][] kp,
    List<ArrayList<String>> data, int baris, int kolom) {
2       double[][] mh = new double[baris][kolom];
3       double kelasI = 0, kelasJ = 0;
4       for (int i = 0; i < baris; i++) {
5           for (int j = 0; j < kolom; j++) {
6               if (data.get(i).get(1).equals("Yes")) {
7                   kelasI = 1;
8               } else {
9                   kelasI = -1;
10              }
11              if (data.get(j).get(1).equals("No")) {
12                  kelasJ = -1;
13              } else {
14                  kelasJ = 1;
15              }
16              mh[i][j] = kelasI * kelasJ * (kp[i][j] +
    Math.pow(lambda, ordo));
17          }
18      }
19      for (int i = 0; i < baris; i++) {
20          for (int j = 0; j < kolom-1; j++) {
21              if (mh[i][j] <= mh[i][j + 1]) {
22                  setMaxHessian(mh[i][j + 1]);
23              }
24          }
25      }
26      return mh;
27  }
28
29  public void setMaxHessian(double hessian) {
30      this.maxHessian = hessian;
31  }
32
33  public double getMaxHessian() {
34      return this.maxHessian;
35  }
```

Kode Program 5.8 Matriks Hessian

Keterangan Kode Program 5.8:

Baris 1 : deklarasi *method matriksHessian*() dengan parameter hasil kp, data, jumlah baris dan kolom. *Method* berguna untuk menghitung nilai matriks *Hessian*.

Baris 2 : inisialisasi variabel mh dengan jumlah baris dan kolom sebanyak jumlah.

Baris 3 : inisialisasi variabel kelasI dan kelasJ untuk menyimpan kelas dari setiap dokumen.

Baris 4-18 : melakukan perulangan untuk menghitung matriks Hessian.

Baris 19-25 : melakukan perulangan untuk mencari nilai matrisk Hessian tertinggi untuk perhitungan nilai konstanta *gamma*.

Baris 26 : mengembalikan nilai mh.

Baris 29-31 : deklarasi method setMaxHessian() untuk mengeset nilai maksimal matriks Hessian.

Baris 33-35 : deklarasi method getMaxHessian() untuk mengambil nilai maksimal matriks Hessian.

5.1.3.3 Implementasi Algoritme *Sequential Training* SVM

Perhitungan *sequential training* SVM terbagi atas beberapa tahap yaitu menghitunga nilai E_i, nilai $\delta\alpha$, dan nilai α_i. Potongan kode program untuk proses perhitungan *sequential training* SVM ditunjukan pada Kode Porgram 5.9.

SVM.java

```
1   public double[][][] stSVM(int jumlah, double[][] hessian)
    {
2       double[][][]     stSvm     =     new     double[(int)
    maxIter][3][jumlah];
3       double[] alpha = new double[jumlah];
4       for (int i = 0; i < jumlah; i++) {
5           alpha[i] = 0;
6       }
7       stSvm[0][0] = Ei(hessian, alpha, jumlah);
8       stSvm[0][1] = deltaAlpha(stSvm[0][0], alpha, jumlah);
9       stSvm[0][2] = alphaI(alpha, stSvm[0][1], jumlah);
10      for (int i = 1; i < maxIter; i++) {
11          if (getMaxDA()>=epsilon) {
                stSvm[i][0]  =  Ei(hessian,  stSvm[i  -  1][2],
12  jumlah);
                stSvm[i][1] = deltaAlpha(stSvm[i][0], stSvm[i
13  - 1][2], jumlah);
                stSvm[i][2]   =   alphaI(stSvm[i   -   1][2],
14  stSvm[i][1], jumlah);
                setMaxIter(i);
16          } else{
17              break;
18          }
19      }
20      return stSvm;
21  }
22
23  private double[] Ei(double[][] hessian, double[] alpha, int
    jumlah) {
24      double[] ei = new double[jumlah];
25      for (int i = 0; i < jumlah; i++) {
26          double temp = 0;
27          for (int j = 0; j < jumlah; j++) {
28              temp = temp + (alpha[j] * hessian[i][j]);
29          }
30          ei[i] = temp;
31      }
32      return ei;
33  }
34
35  private double[] deltaAlpha(double[] ei, double[] alpha,
    int jumlah) {
36      double[] deltaAlpha = new double[jumlah];
37      for (int i = 0; i < jumlah; i++) {
38          deltaAlpha[i]  =  Math.min(Math.max(gamma  *  (1  -
    ei[i]), (-alpha[i])), c - alpha[i]);
39      }
40      for (int i = 0; i < jumlah - 1; i++) {
41          if (deltaAlpha[i] <= deltaAlpha[i + 1]) {
42              setMaxDA(deltaAlpha[i+1]);
43          }
44      }
45      return deltaAlpha;
46  }
47
48  private    double[]    alphaI(double[]    alpha,    double[]
    deltaAlpha, int jumlah) {
49      double[] alphaI = new double[jumlah];
50      for (int i = 0; i < jumlah; i++) {
```

```
51              alphaI[i] = alpha[i] + deltaAlpha[i];
52          }
53          return alphaI;
54      }
55
56      public void setMaxDA(double da) {
57          this.maxDA = da;
58      }
59
60      public double getMaxDA() {
61          return this.maxDA;
62      }
63
64      public void setMaxIter(int i){
65          this.iterasi = i;
66      }
67
68      public int getMaxIter(){
69          return this.iterasi;
70      }
```

Kode Program 5.9 Sequential Training SVM

Keterangan Kode Program 5.9:

Baris 1-21 : deklarasi *method stSVM*() dengan parameter jumlah dokumen dan hasil mh. *Method* berguna untuk menghitung *sequential training* SVM.

Baris 2 : inisialisasi variable stSvm berguna menampung hasil perhitungan nilai *ei*, *delta alpha*, dan *alpha-i*.

Baris 3 : inisialisasi variable *alpha* dengan jumlah baris sebanyak jumlah.

Baris 4-6 : melakukan perulangan untuk mengeset nilai alpha menjadi 0.

Baris 7 : memanggil *method* Ei() untuk menghitung nilai *ei* dengan parameter hasil mh, alpha dan jumlah.

Baris 8 : memanggil *method deltaAlpha*() untuk menghitung nilai delta alpha dengan parameter nilai *ei*, *alpha* dan jumlah.

Baris 9 : memanggil *method alphaI*() untuk menghitung nilai *alpha* ke-I dengan parameter nilai *alpha*, *deltaAlpha* dan jumlah.

Baris 10-19 : melakukan perulangan untuk menghitung nilai *ei*, *deltaAlpha* dan *alphaI* sebayak jumlah maksimal iterasi dengan kondisi jika nilai maksimal dari *deltaAlpha* kurang dari nilai *epsilon* maka perulangan berhenti.

Baris 20 : mengembalikan nilai stSvm.

Baris 23-33 : deklarasi *method Ei*() dengan parameter hasil *hessian*, *alpha* dan jumlah. *Method* berguna untuk menghitung nilai *ei*.

Baris 24 : inisialisasi variabel *ei* dengan jumlah baris sebanyak jumlah dokumen.

Baris 25-31 : melakukan perulangan untuk menghitung nilai *ei*.

Baris 32 : mengembalikan nilai *ei*.

Baris 35-46 : deklarasi *method deltaAlpha*() dengan parameter nilai *ei*, *alpha* dan jumlah. *Method* berguna menghitung nilai *deltaAlpha*.

Baris 36 : inisialisasi variabel *deltaAlpha* dengan jumlah baris sebanyak jumlah dokumen.

Baris 37-39 : melakukan perulangan untuk menghitung nilai *deltaAlpha*.

Baris 40-44 : melakukan perulangan untuk mencari nilai *deltaAlpha* maksimal.

Baris 45 : mengembalikan nilai *deltaAlpha*.

Baris 48-54 : deklarasi *method alphaI()* dengan parameter nilai *alpha*, *deltaAlpha* dan jumlah. *Method* berguna untuk menghitung nilai *alpha* ke-i.

Baris 49 : inisialisasi variabel *alphaI* dengan jumlah baris sebanyak jumlah dokumen.

Baris 50-52 : melakukan perulangan untuk menghitung nilai *alphaI*.

Baris 53 : mengembalikan nilai *alphaI*.

Baris 56-58 : deklarasi *method setMaxDA*() yang berguna untuk menyimpan nilai *deltaAlpha* maksimal.

Baris 60-62 : deklarasi *method getMaxDA*() yang berguna untuk mengambil nilai *deltaAlpha* maksimal.

Baris 64-66 : deklarasi *method setIterasi*() yang berguna untuk menyimpan jumlah iterasi yang dilakukan.

Baris 68-70 : deklarasi *method getIterasi*() yang berguna untuk mengambil nilai iterasi.

Proses selanjutnya yaitu perhitungan *Support Vector*. Perhitungan *Support Vector* berguna untuk mendapatkan model SV dan bias yang digunakan untuk *testing* data. Potongan kode program untuk proses perhitungan *Support Vector* ditunjukan pada Kode Porgram 5.10.

SVM.java

```
1   public double[][] supportVector(double[] alpha,
    List<ArrayList<String>> data, int jumlah) {
2       double[][] sv = new double[jumlah][2];
3       for (int i = 0; i < jumlah; i++) {
4           sv[i][0] = alpha[i];
5           if (data.get(i).get(1).equals("Yes")) {
6               sv[i][1] = 1;
7           } else {
8               sv[i][1] = -1;
9           }
10          if (data.get(i).get(1).equals("No")) {
11              sv[i][1] = -1;
12          } else {
13              sv[i][1] = 1;
14          }
15      }
16      return sv;
17  }
18
19  public void minMaxSV(double[] alpha, int jumlah,
    double[][] sv) {
20      maxSV = new double[1][2];
21      minSV = new double[1][2];
22      maxSV[0][0] = alpha[0];
23      maxSV[0][1] = sv[0][1];
24      minSV[0][0] = alpha[jumlah/2];
25      minSV[0][1] = sv[jumlah/2][1];
26      kelasMin = jumlah/2;
27      for (int i = 0; i < jumlah; i++) {
28          if (sv[i][1] == 1) {
29              if (maxSV[0][0] < alpha[i]) {
30                  maxSV[0][0] = alpha[i];
31                  maxSV[0][1] = sv[i][1];
32                  kelasMax = i;
33              }
34          }
35          if (sv[i][1] == -1) {
36              if (minSV[0][0] < alpha[i]) {
37                  minSV[0][0] = alpha[i];
38                  minSV[0][1] = sv[i][1];
39                  kelasMin = i;
40              }
41          }
42      }
43  }
44
45  public double[][] getMaxSV() {
46      return maxSV;
47  }
48
49  public double[][] getMinSV() {
50      return minSV;
51  }
52
53  public double hitungBias(double[][] kp, double[][] sv, int
    jumlah) {
54      double[] max = new double[jumlah];
55      double[] min = new double[jumlah];
56      double[] maxA = new double[jumlah];
```

```
57      double[] minA = new double[jumlah];
58      double jumlahMax = 0, jumlahMin = 0;
59      for (int i = 0; i < jumlah; i++) {
60          max[i] = kp[kelasMax][i];
61          min[i] = kp[kelasMin][i];
62      }
63      for (int i = 0; i < jumlah; i++) {
64          maxA[i] = max[i] * sv[i][0] * sv[i][1];
65          minA[i] = min[i] * sv[i][0] * sv[i][1];
66          jumlahMax += maxA[i];
67          jumlahMin += minA[i];
68      }
69      double bias = -0.5 * (jumlahMax + jumlahMin);
70      return bias;
71  }
```

Kode Program 5.10 Support Vector

Keterangan Kode Program 5.10:

Baris 1-17 : deklarasi *method supportVector*() dengan parameter *alpha* terakhir, data dan jumlah dokumen. *Method* berguna untuk menghitung nilai *support vector*.

Baris 2 : inisialisasi variable sv dengan jumlah baris sebanyak jumlah dan kolom sebanyak 2 untuk menampung *support vector* dari masing-masing kelas.

Baris 3-15 : melakukan perulangan untuk mencari nilai kelas pada setiap dokumen.

Baris 16 : mengembalikan nilai sv.

Baris 19-43 : deklarasi *method minMaxSV*() dengan parameter nilai *alpha* terakhir, jumlah dokumen dan nilai sv. Method berguna untuk mencari nilai maximal dan minimal pada masing-masing kelas.

Baris 20-21 : inisialisasi variable maxSV dan minSV untuk menampung nilai *alpha* dan kelas sv.

Baris 22-25 : menyimpan nilai *alpha* dan sv ke variable maxSV dan minSV.

Baris 26 : menyimpan nilai kelasMin.

Baris 27-42 : melakukan perulangan untuk mencari nilai maksimal dan minimal pada masing-masing kelas.

Baris 45-47 : deklarasi *method getMaxSV*() untuk mengembalikan nilai variabel maxSV.

Baris 49-51 : deklarasi *method getMinSV*() untuk mengembalikan nilai variabel minSV.

Baris 53-71 : deklarasi *method hitungBias*() dengan parameter nilai hessian, nilai sv dan jumlah dokumen. *Method* berguna untuk menghitung nilai bias.

Baris 54-58 : inisialisasi variable *max* dan *min* untuk menampung nilai maksimal masing-masing kelas, dan inisialisasi variable maxA dan minA untuk menampung nilai perkalian *max* dan *min* dengan *alpha*.

Baris 59-62 : melakukan perulangan untuk mengambil nilai matriks *Hessian* dari hasil nilai *min-max support* vector.

Baris 63-68 : melakukan perulangan untuk menghitng nilai *min-max Hessian* dikalikan dengan alpha kemudian dijumlahkan.

Baris 69 : menghitung nilai bias.

Baris 70 : mengembalikan nilai bias.

BAB 6 PEMBAHASAN

Pada bab ini membahasan tentang analisis dari hasil implementasi sistem dengan melakukan beberapa pengujian. Pengujian dilakukan untuk mengetahui pengaruh tiap parameter pada hasil akurasi identifikasi sistem. Pengujian yang dilakukan yaitu pengujian parameter SVM dan *threshold* IG.

6.1 Pengujian Paramater Sequential Training SVM

Pengujian yang dilakukan pada parameter *sequential training* SVM terdapat 5 skenario pengujian dengan 10 nilai berbeda yang dilakukan sebanyak 1 kali percobaan karena dalam perhitungan data tidak ada nilai yang *random*. Skenario pengujian yang dilakukan yaitu pengujian nilai *lambda*, konstanta *gamma*, nilai *epsilon*, pengaruh nilai C, dan maksimum iterasi.

Data yang digunakan pada pengujian sebanyak 300 *tweet* dimana data *tweet bully* sebanyak 150 dan bukan *bully* sebanyak 150, maka data yang digunakan menjadi seimbang. Skenario perbandingan yang digunakan adalah 80:20, dimana jumlah data *training* sebanyak 240 *tweet* dan data *testing* 60 *tweet* dengan jumlah *bully* dan bukan *bully* yang seimbang.

Pada pengujian ini menggunakan hasil *Accuracy*, *Precision*, *Recall* dan *F-measure*. Hal ini dikarenakan pada penelitian ini hanya mengidentifikasi *tweet* yang mengandung *cyberbullying*, sehingga *retrieve* hasil identifikasi dibutuhkan untuk mengetahui apakah informasi yang diminta oleh pengguna sudah sesuai dengan informasi yang diberikan oleh sistem.

6.1.1 Skenario Pengujian Nilai Lambda

Pengujian pada parameter *sequential training* SVM dilakukan untuk mengetahui pada skenario mana mendapatkan nilai terbaik untuk nilai λ (*lambda*). Nilai parameter sequential training SVM yang digunakan dalam pengujian adalah ϒ = 0.001, ε = 0.0001, C = 1, dan iterMax = 100. Hasil pengujian terhadap nilai *lambda* ditunjukan pada Tabel 6.1.

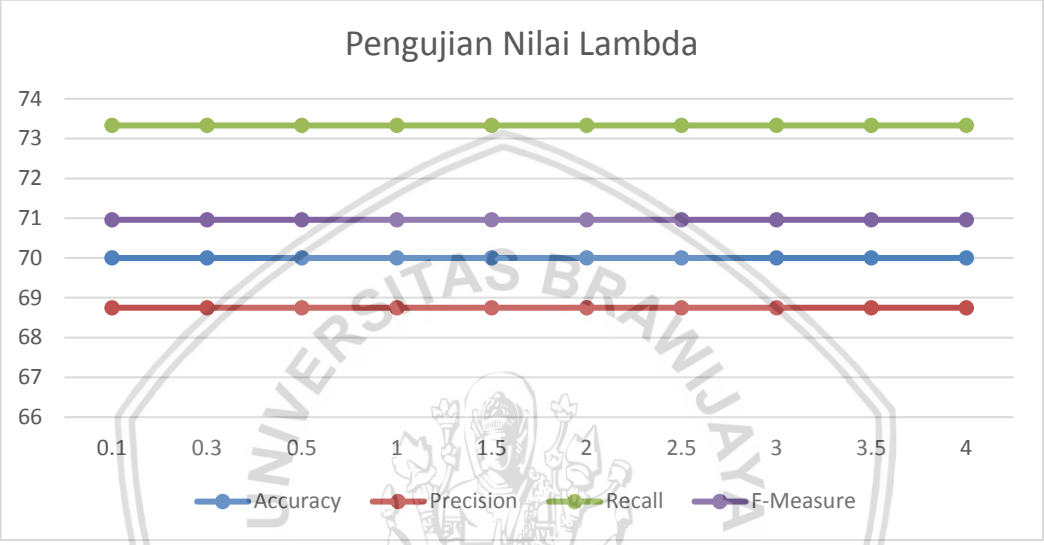
Tabel 6.1 Hasil Pengujian Nilai *Lamda*

Lambda	Hasil			
	Accuracy	Precision	Re-call	F-Measure
0.1	70	68.75	73.33	70.96
0.3	70	68.75	73.33	70.96
0.5	70	68.75	73.33	70.96
1	70	68.75	73.33	70.96
1.5	70	68.75	73.33	70.96
2	70	68.75	73.33	70.96
2.5	70	68.75	73.33	70.96

3	70	68.75	73.33	70.96
3.5	70	68.75	73.33	70.96
4	70	68.75	73.33	70.96

6.1.2 Analisis Pengujian Nilai Lambda

Dari data hasil pengujian Tabel 6.1 diubah menjadi bentuk grafik, hasil analisis pengujian *lambda* dalam bentuk grafik yang ditunjukan pada Gambar 6.1.



Gambar 6.1 Grafik Hasil Pengujian Nilai Lambda

Berdasarkan hasil pengujian yang telah dilakukan dan diuraikan pada grafik Gambar 6.1 didapatkan bahwa hasil yang didapatkan adalah konstan pada semua nilai lambda yang diujikan yaitu *accuracy* 70%, *precision* 68.75%, *recall* 73.33% dan *f-measure* 70.96%. Hal ini terjadi karena nilai *lambda* hanya digunakan untuk melakukan perhitungan matriks *hessian*. Dan matriks *hessian* digunakan ketika menghitung nilai *Ei* untuk *sequential training* SVM. Sehingga hasil yang didapatkan tidak terlalu mempengaruhi nilai *bias*.

6.1.3 Skenario Pengujian Nilai Konstanta Gamma

Pengujian pada parameter *sequential training* SVM dilakukan untuk mengetahui pada skenario mana mendapatkan nilai terbaik untuk nilai ϒ (konstanta *gamma*). Nilai parameter *sequential training* SVM yang digunakan dalam pengujian adalah λ = 0.5, ε = 0.0001, C = 1, dan iterMax = 100. Hasil pengujian nilai konstanta *gamma* ditunjukan pada Tabel 6.2.

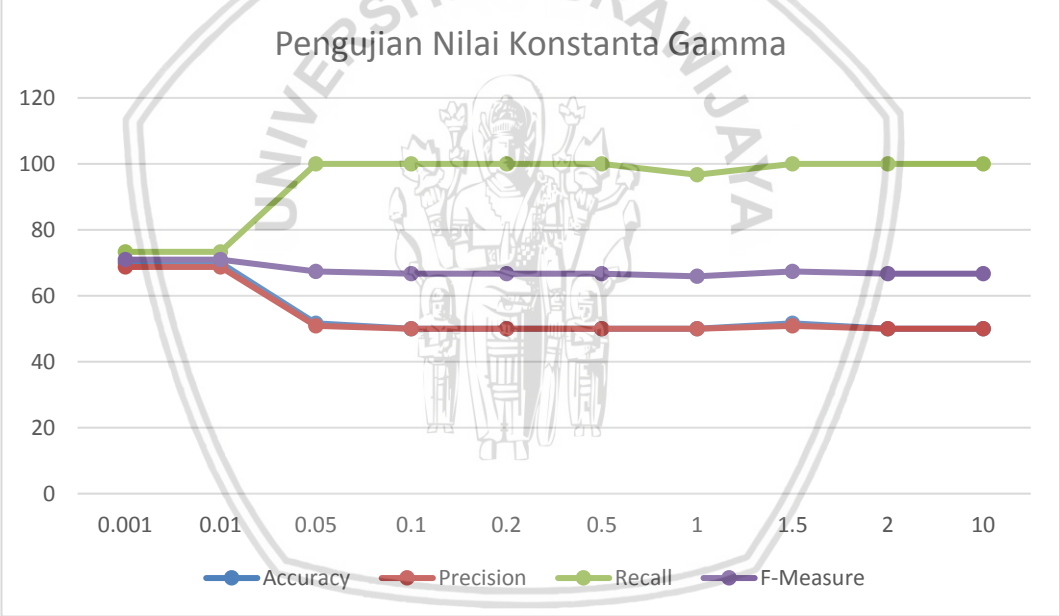
Tabel 6.2 Hasil Pengujian Nilai Konstanta *Gamma*

Konstanta Gamma	Hasil			
	Accuracy	Precision	Re-call	F-Measure
0.001	70	68.75	73.33	70.96

0.01	70	68.75	73.33	70.96
0.05	51.66	50.84	100	67.41
0.1	50	50	100	66.66
0.2	50	50	100	66.66
0.5	50	50	100	66.66
1	50	50	96.66	65.9
1.5	51.66	50.84	100	67.41
2	50	50	100	66.66
10	50	50	100	66.66

6.1.4 Analisis Pengujian Nilai Konstanta *Gamma*

Dari data hasil pengujian Tabel 6.2 pengujian nilai konstanta *gamma* maka dianalisis ke dalam bentuk grafik yang ditunjukan pada Gambar 6.2.



Gambar 6.2 Grafik Hasil Pengujian Nilai Konstanta Gamma

Berdasarkan hasil pengujian yang diuraikan pada grafik Gambar 6.2 didapatkan bahwa hasil terbaik didapatkan pada nilai *gamma* 0.001 dan 0.01, kemudian nilai menurun ketika nilai *gamma* semakin tinggi. Hal in disebabkan karena nilai *gamma* digunakan untuk menghitung nilai *delta alpha*, dimana nilai *delta* alpha merupakan nilai yang menentukan apakah hasil konvergen atau belum. Nilai *gamma* didapatkan dengan cara membagi nilai konstanta *gamma* dengan nilai maksimal matriks *Hessian*. Ketika nilai *gamma* tinggi maka hasil lompatannya akan tinggi juga sehingga perhitungan semakin cepat konvergen. Nilai *gamma* yang diambil adalah 0.001 dengan *accuracy* 70%, *precision* 68.75%, *recall* 73.33% dan *f-measure* 70.96%.

6.1.5 Skenario Pengujian Nilai *Epsilon*

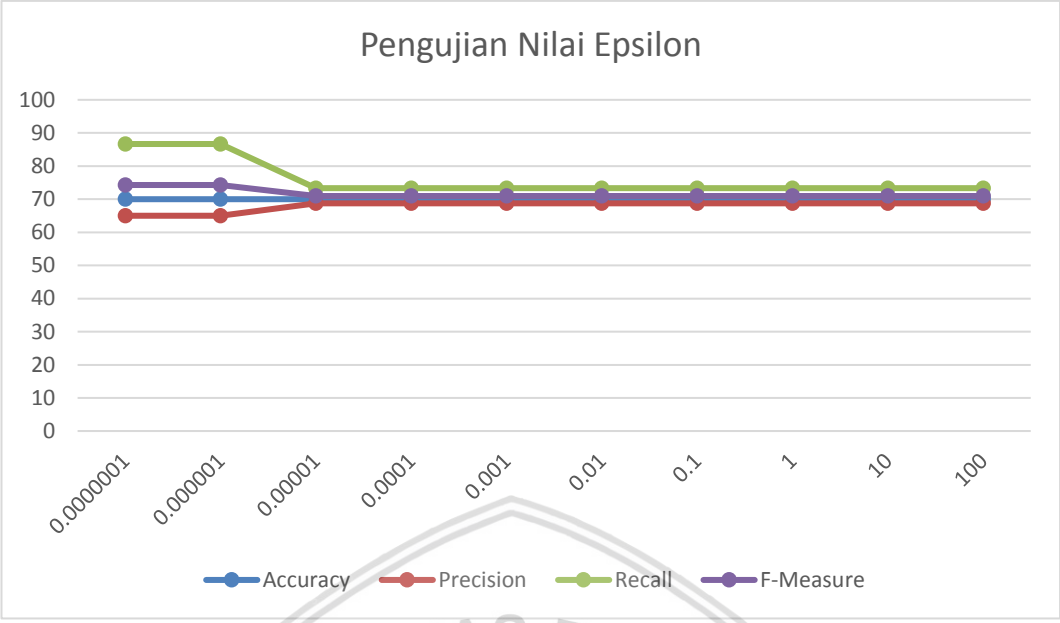
Pengujian pada parameter *sequential training* SVM dilakukan untuk mengetahui pada skenario mana mendapatkan nilai terbaik untuk nilai ε (*epsilon*). Nilai parameter *sequential training* SVM yang digunakan dalam pengujian adalah λ = 0.5, ϒ = 0.001, C = 1, dan iterMax = 100. Hasil pengujian terhadap nilai *epsilon* ditunjukan pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Nilai *Epsilon*

Epsilon	Hasil			
	Accuracy	Precision	Re-call	F-Measure
0.0000001	68.33	64.10	83.33	72.46
0.000001	68.33	64.10	83.33	72.46
0.00001	70	68.75	73.33	70.96
0.0001	70	68.75	73.33	70.96
0.001	70	68.75	73.33	70.96
0.01	70	68.75	73.33	70.96
0.1	70	68.75	73.33	70.96
1	70	68.75	73.33	70.96
10	70	68.75	73.33	70.96
100	70	68.75	73.33	70.96

6.1.6 Analisis Pengujian Nilai *Epsilon*

Dari data hasill pengujian Tabel 6.3 pengujian nilai *epsilon* maka dianalisis ke dalam bentuk grafik yang ditunjukan pada Gambar 6.3. Berdasarkan hasil pengujian yang telah dilakukan pada grafik Gambar 6.3 didapatkan bahwa hasil terbaik pada nilai *epsilon* kurang dari sama dengan 0.000001 dan 0.0000001. Hal ini disebabkan karena nilai *epsilon* digunakan sebagai batas maksimal untuk hasil konvergen. Ketika nilai *epsilon* semakin tinggi, maka hasil akan semakin cepat konvergen. Pemilihan nilai epsilon 0.000001 dengan *accuracy* 68.33%, *precision* 64.10%, *recall* 83.33% dan *f-measur*e 72.46%, karena jika nilai epsilon yang dipilih terlalu rendah maka lompatan perhitungan yang dilakukan kecil sehingga untuk mencapai nilai konvergen akan lama, sedangkan jika nilai epsilon yang dipilih terlalu tinggi maka lompatan perhitungan yang dilakukan besar sehingga lebih cepat mencapai nilai konvergen. Maka dari itu nilai epsilon yang dipilih adalah yang ditengah-tengah.



Gambar 6.3 Grafik Hasil Pengujian Nilai Epsilon

6.1.7 Skenario Pengujian Maksimum Iterasi

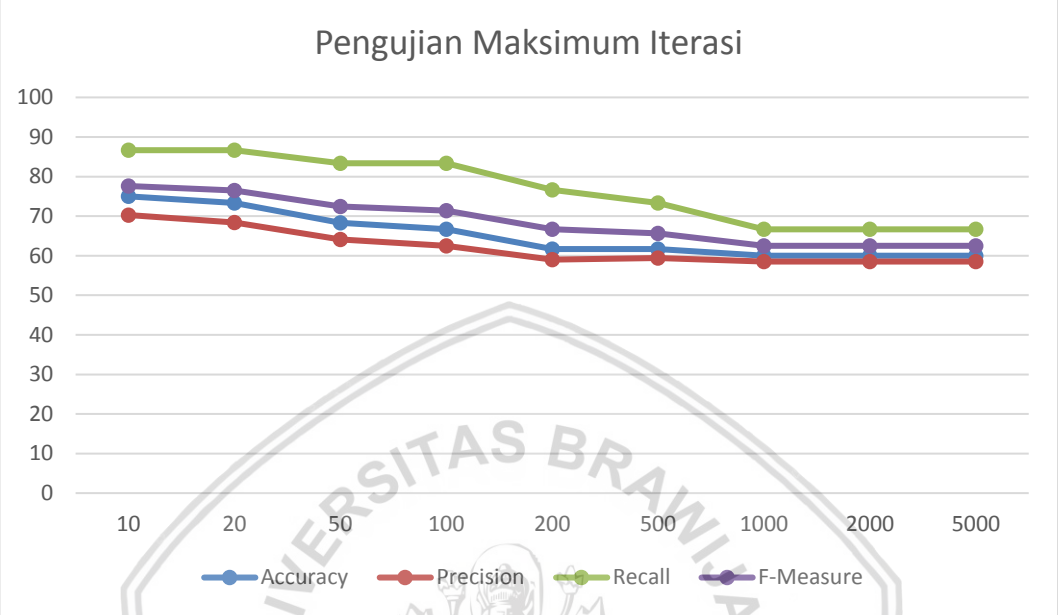
Pengujian pada parameter *sequential training* SVM dilakukan untuk mengetahui pada skenario mana mendapatkan nilai terbaik untuk nilai maksimum iterasi (iterMax). Nilai parameter *sequential training* SVM yang digunakan dalam pengujian adalah λ = 0.5, ϒ = 0.001, ε = 0.000001, dan C = 1. Hasil pengujian terhadap nilai maksimum iterasi ditunjukan pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian Nilai Maksimum Iterasi

Iterasi	Hasil			
	Accuracy	Precision	Re-call	F-Measure
10	71.66	67.56	83.33	74.62
20	75	70.27	86.66	77.61
50	73.33	68.42	86.66	76.47
100	68.33	64.1	83.33	72.46
200	66.66	62.5	83.33	71.42
500	61.66	58.97	76.66	66.66
1000	61.66	59.45	73.33	65.67
2000	60	58.82	66.66	62.5
5000	60	58.82	66.66	62.5
10000	60	58.82	66.66	62.5

6.1.8 Analisis Pengujian Maksimum Iterasi

Dari data hasil pengujian Tabel 6.4 diubah menjadi bentuk grafik agar lebih mudah dianalisis. Berikut hasil analisis pengujian nilai maksimum iterasi dalam bentuk grafik yang ditunjukan pada Gambar 6.4.



Gambar 6.4 Grafik Hasil Pengujian Nilai Maksimum Iterasi

Berdasarkan pengujian yang telah dilakukan dan diuraikan pada grafik Gambar 6.4 bahwa hasil terbaik yang didapatkan terdapat pada iterasi sebanyak 20 dengan nilai *accuracy* 75%, *precision* 70.27%, *recall* 86.66% dan *f-measur*e 77.61%. Kemudian pada iterasi lebih dari 1000 hasil yang dihasilkan sama, hal ini dikarenakan perhitungan telah memasuki keadaan konvergen pada iterasi ke-1404.

Pada saat kondisi konvergen hasil identifikasi *tweet cyberbullying* menjadi menurun dapat disebabkan oleh keadaan dimana data *training* yang dibuat tidak mewakilkan seluruh data yang akan digunakan pada *testing* karena data yang digunakan tidak cukup untuk sepenuhnya menentukan penggolongan, atau biasa disebut *underfitting* (Domingos, 2012). Keadaan ini dapat menyebabkan performa yang buruk dalam melatih data. Karena pada objek data yang digunakan yaitu *tweet* tentang Bapak Fadli Zon sangat luas, sehingga pada setiap dokumen memiliki isi yang berbeda-beda dan dapat menghasilkan varian yang banyak.

6.1.9 Skenario Pengujian Pengaruh Nilai C

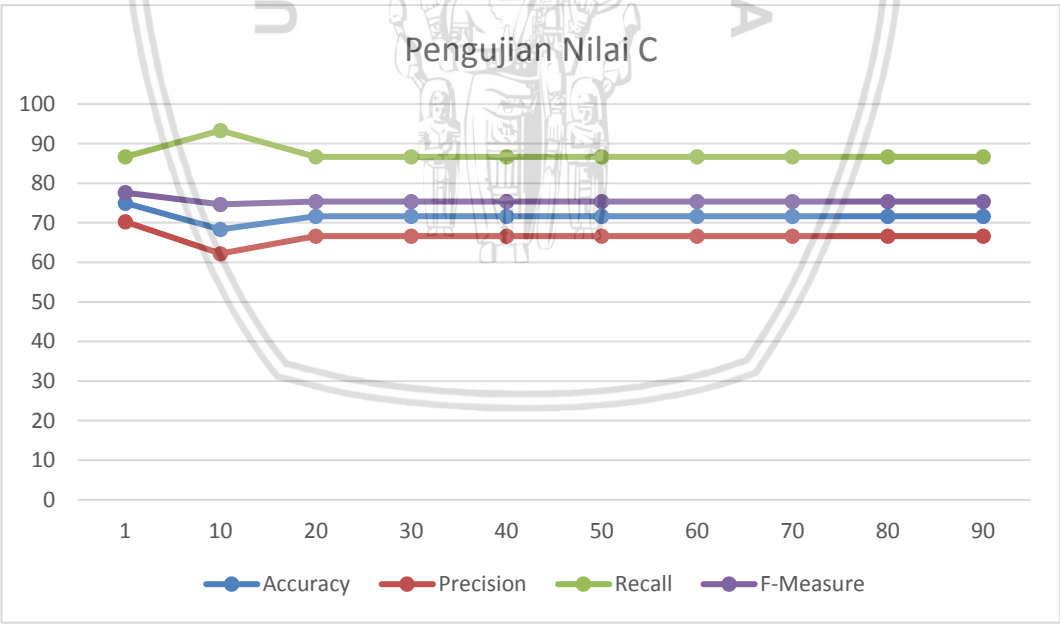
Pengujian pada parameter *sequential training* SVM dilakukan untuk mengetahui pada skenario mana mendapatkan nilai terbaik untuk nilai C (*Complexity*). Nilai parameter *sequential training* SVM yang digunakan dalam pengujian adalah $\lambda = 0.5$, $\Upsilon = 0.001$, $\varepsilon = 0.000001$, dan iterMax = 20. Hasil pengujian terhadap nilai C ditunjukan pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian Nilai C

C	Hasil			
	Accuracy	Precision	Re-call	F-Measure
1	75	70.27	86.66	77.61
10	68.33	62.22	93.33	74.66
20	71.66	66.66	86.66	75.36
30	71.66	66.66	86.66	75.36
40	71.66	66.66	86.66	75.36
50	71.66	66.66	86.66	75.36
60	71.66	66.66	86.66	75.36
70	71.66	66.66	86.66	75.36
80	71.66	66.66	86.66	75.36
90	71.66	66.66	86.66	75.36

6.1.10 Analisis Pengujian Pengaruh Nilai C

Dari data hasil pengujian Tabel 6.5 pengujian nilai C maka dianalisis ke dalam bentuk grafik yang ditunjukan pada Gambar 6.5.



Gambar 6.1 Grafik Hasil Pengujian Nilai C

Berdasarkan dari hasil pengujian grafik Gambar 6.5 didapatkan bahwa hasil terbaik pada pengujian nilai C sama dengan 1 dengan nilai *accuracy* 75%, *precision* 70.27%, *recall* 86.66% dan *f-measur*e 77.61%. Hal ini disebabkan karena semakin tinggi nilai C maka nilai kernel polynomial akan semakin tinggi pula. Ketika nilai *kernel polynomial* semakin tinggi maka nilai matriks *Hessian* semakin tinggi pula, sehingga menyebabkan pada perhitungan nilai konstanta *gamma* yang hasilnya

didapatkan semakin kecil kemudian dapat menyebabkan iterasi menjadi semakin cepat konvergen.

6.2 Pengujian Paramater *Threshold* IG

Pengujian yang dilakukan untuk parameter *threshold* seleksi fitur IG terdapat 1 skenario pengujian dengan 10 nilai *threshold* yang berbeda.

6.2.1 Skenario Pengujian *Threshold* IG

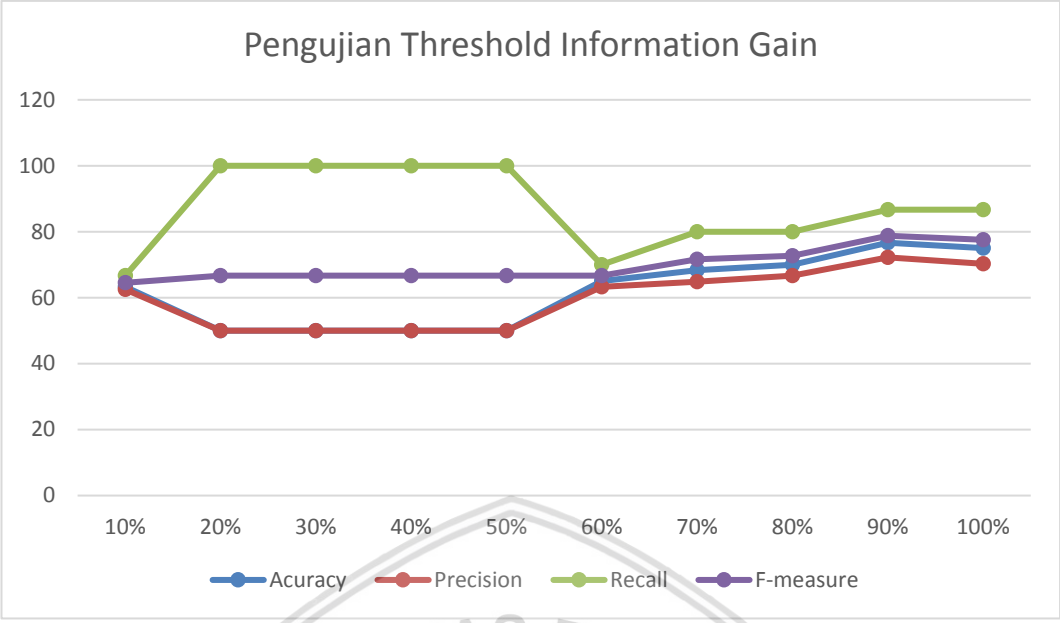
Pengujian pada parameter *threshold* IG dilakukan untuk mengetahui pada skenario mana mendapatkan nilai terbaik untuk nilai *threshold*. Hasil pengujian terhadapt nilai *threshold* IG ditunjukan pada Tabel 6.6.

Tabel 6.6 Hasil Pengujian *Threshold* IG

Threshold	Hasil			
	Accuracy	Precision	Re-call	F-Measure
10%	63.33	62.5	66.66	64.51
20%	50	50	100	66.66
30%	50	50	100	66.66
40%	50	50	100	66.66
50%	50	50	100	66.66
60%	65	63.63	70	66.66
70%	68.33	64.86	80	71.64
80%	70	66.66	80	72.72
90%	76.66	72.22	86.66	78.78
100%	75	70.27	86.66	77.61

6.2.2 Analisis Pengujian *Threshold* IG

Dari data hasil pengujian Tabel 6.6 pengujian nilai *threshold* seleksi fitur *information gain* maka dianalisis ke dalam bentuk grafik yang ditunjukan pada Gambar 6.6.



Gambar 6.2 Grafik Hasil Pengujian Threshold IG

Berdasarkan hasil pengujian pada grafik Gambar 6.6 hasil terbaik didapatkan pada nilai *threshold* 90%. Ketika semua fitur digunakan hasil yang didapatkan *accuracy* sebesar 76.66%, *precision* 72.22%, *recall* 86.66% dan *f-measure* 78.78%. Dan ketika fitur yang digunakan sebanyak 90% nilai akurasi, *precision*, *recall* dan *f-measure* menjadi meningkat karena fitur yang relevan pada kedua kelas tidak digunakan sehinggi nilai akurasi menjadi meningkat. Dan ketika nilai *threshold* yang digunakan semakin kecil maka hasil akurasi menurun dibanding dengan *threshold* 90% karena fitur yang digunakan semakin sedikit dan tidak semua fitur yang unik berada pada data *testing*.

Pada saat pemilihan *term* menggunakan seleksi fitur *information gain* proses menghasilkan nilai yang tinggi pada *term* yang unik dimana *term* tersebut muncul sekali pada satu kelas. Nilai *information gain* akan mempengaruhi *term* yang digunakan pada proses identifikasi, dimana *term* dengan nilai *information gain* yang tinggi yang akan digunakan. Misalkan pada nilai *threshold* 100%, *term* "indonesia" merupakan fitur yang relevan dalam kedua kelas, jadi ketika fitur tersebut masuk maka nilai akurasi akan menurun. Kemudian misal pada *threshold* 90%, *term* "indonesia" dihapus sehingga jumlah fitur yang tersisa akan relevan dengan data *testing* dan dapat meningkatkan akurasi.

BAB 7 PENUTUP

Pada bab ini berisi kesimpulan dan saran dari hasil penelitian yang telah dilakukan. Kesimpulan yang diambil merupakan hasil analisis yang telah dilakukan berdasarkan pengujian sistem, dan saran ditunjukan untuk penelitian yang selanjutnya.

7.1 Kesimpulan

Berdasarkan hasil penelitian, pengujian dan anilisis yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Hasil pengujian identifikasi *tweet cyberbullying* menggunakan metode SVM mendapatkan hasil terbaik berdasarkan pengujian *iterMax*, parameter λ (*lambda*), γ (konstanta *gamma*), ε (*epsilon*), dan C (*complexity*), pada *sequential training* SVM berpengaruh pada perubahan nilai bobot α_i (*alpha* ke-*i*) dan nilai *b* (*bias*). Hasil terbaik yang didapatkan dari seluruh pengujian parameter *sequential training* SVM yaitu *iterMax* = 20, λ = 0.5, γ = 0.001, ε = 0.000001, dan C = 1. Dan hasil akurasi yang diperoleh yaitu *accuracy* 75%, *precision* 70.27%, *recall* 86.66% dan *f-measur*e 77.61%.
2. Berdasarkan hasil pengujian pada *threshold* seleksi fitur *information gain*, hasil terbaik didapatkan adalah pada nilai *threshold* 90%, dengan nilai *accuracy* 76.66%, *precision* 72.22%, *recall* 86.66% dan *f-measure* 78.78%. Hal ini disebabkan karena seleksi fitur *information gain* mempunyai nilai yang tinggi jika fitur tersebut merepresentasikan suatu kelas tertentu, dan mempunyai nilai yang rendah jika fitur tersebut muncul didalam semua kelas. Jadi hasil identifikasi *tweet cyberbullying* dengan seleksi fitur mendapatkan akurasi lebih tinggi dibandingkan menggunakan seluruh fitur yang ada.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan, maka saran yang dapat diberikan untuk pengembangan penelitian selanjutnya, yaitu:

1. Penelitian selanjutnya diharapkan menambah jumlah data yang digunakan, karena akan berpengaruh pada proses identifkasi untuk mendapatkan hasil yang optimal.
2. Pada penelitian selanjutnya perlu mempertimbangkan *dimensionality* dari data *training*. Karena jika memiliki data *training* yang *dimensionality* tinggi kemungkinan dapat menyebabkan keadaan *overfitting* dan *underfitting*.

DAFTAR PUSTAKA

Adriansyah, M. et al., 2017. Cyberbullying Comment Classification on Indonesian Selebgram Using Support Vector Machine Method. *Research Gate.*

Agusta, L., 2009. Perbandingan Algoritma Stemming Porter dengan Algoritma Nazief & Adriani untuk Stemming Dokumen Teks Bahasa Indonesia. *Konferensi Nasional Sistem dan Informatika,* p. 2.

Bangsheng, S., 2013. *Information Gain Feature Selection Based on Feature Interactions.* Houston: s.n.

Chandani, V., Wahono, R. S. & Purwanto, 2015. Komparasi ALgoritma Klasifikasi Machine Learning dan Feature Selection pada Analisis Sentimen Review Film. *Journal of Intellignet Systems,* Volume 1.

Darma, I. M. B. S., 2018. Penerapan Sentimen Analisis acara Televisi pada Twitter Menggunakan Support Vector Machine dan Algoritma Genetika sebagai Metode Seleksi Fitur. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2.

Devaga, Evita. 2017. RI Perangi Konten Negatif. Tersedia di: <https://www.kominfo.go.id/content/detail/11226/ri-perangi-konten-negatif/0/sorotan_media> [Diakses 26 Januari 2018].

Deng, H. & Runger, G., 2012. Feature Selection via Regularized Trees. *The 2012 International Joint Conference on Neural Networks (IJCNN),* Volume 3.

Domingos, P., 2012. A Few Useful Things to Know About Machine Learning. *Communication Of The ACM,* Volume 55, p. 4.

Gaigole, P. C., Patil, L. H. & Chaudhari, P. M., 2013. *Preprocessing Techniques in Text Categorization.* s.l., International Journal of Computer Application.

Garcia, E. 2005. *Document Indexing Tutorial*. Tersedia di: <http://www.miislita.com/information-retrieval-tutorial/indexing.html> [Diakses 29 September 2017].

Motivasee, 2017. Apa itu Twitter dan Tips Cara Menggunakannya. Motivasee. Tersedia di: <http://motivasee.com/twitter/> [Diakses 29 September 2017].

Noviantho, Isa, S. M. & Ashianti, L., 2017. Cyberbullying Classification using Text Mining. *IEEE.*

Nugroho, A. S., Witarto, A. B. & Handoko, D., 2003. Teori dan Aplikasinya dalam Bioinformatika. *IEEE.*

Nurmansyah. 2015. R, Bahasa Pemrograman untuk Analisis Data dan Statistik. Kompasiana. Tersedia di: <https://www.kompasiana.com/nurmansyah/r-bahasa-pemrograman-untuk-analisis-data-dan-statistik_552fab4f6ea8349b138b456f> [Diakses 26 Januari 2018].

Oktaviani, Kirana. 2013. Apa itu *Cyber Bullying*. Kompasiana. Tersedia di: <https://www.kompasiana.com/kiranaoktaviani/apa-itu-cyber-bullying_552ff83a6ea8344b778b45d4> [Diakses 29 September 2017].

Patchin, J. & Hinduja, S., 2012. *Cyberbullying Prevention and Response: Expert Perspectives.* New York: Emotional and Behavioural Difficulties.

Putubuku. 2008. *Recall & Precision*. Ilmu Perustakaan & Informasi. Tersedia di: < https://iperpin.wordpress.com/2008/03/27/recall-precision/> [Diakses 26 Januari 2018].

Putranti, N. D. & Winarko, E., 2014. Analisis Sentimen Twitter untuk Teks Berbahasa Indonesi dengan Maximum Entropy dan Support Vector Machine. *IJCCS,* Volume 8.